PRINCÍPIOS DE BIOESTATÍSTICA E EPIDEMIOLOGIA

Fundamentos práticos em R

LUCIANO GAMA DA SILVA GOMES

PRINCÍPIOS DE

BIOESTATÍSTICA E EPIDEMIOLOGIA

Fundamentos práticos em R

LUCIANO GAMA DA SILVA GOMES

Título original
Princípios de bioestatística e epidemiologia –
Fundamentos práticos em R

Primeira publicação em
Salvador, Bahia, Brasil.
2025

1ª Edição

Todos os direitos da obra
Princípios de bioestatística e epidemiologia –
Fundamentos práticos em R
reservados ao Autor

Copyright do texto © Luciano Gama da Silva Gomes, 2025
Arte da capa – o Autor

Dados Internacionais de Catalogação na Publicação (CIP)

(Câmara Brasileira do Livro, SP, Brasil)

Gomes, Luciano Gama da Silva
Princípios de bioestatística e epidemiologia : fundamentos práticos em R / Luciano Gama da Silva Gomes. -- 1. ed. -- Salvador, BA : Ed. do Autor, 2025.

Bibliografia.
ISBN 978-65-01-50717-0

1. Bioestatística 2. Educação em saúde 3. Epidemiologia 4. Linguagem de programação para computadores I. Título.

25-276793 CDD-570.15195

Índices para catálogo sistemático:

1. Bioestatística para ciências da saúde 570.15195

Aline Graziele Benitez - Bibliotecária - CRB-1/3129

Sobre o autor

Luciano formou-se biomédico na Universidade Estadual de Santa Cruz (Bahia, Brasil), onde iniciou a vida acadêmica. Fez mestrado em Genética e Biologia Molecular pela mesma Universidade e desde então trabalha muito com a bioestatística, decidindo realizar seu estágio de docência em tal disciplina. Tem especialização em Ciência de Dados e Big Data Analytis (Faculdade Metropolitana) e em Gestão em Saúde (Universidade Federal do Recôncavo da Bahia). É doutor em Imunologia pela Universidade Federal da Bahia, onde estudou a genética da asma através de modelos preditivos de inteligência artificial. Fez pós-doutorado na The London School of Hygiene & Tropical Medicine, vinculado ao Department of Medical Statistics, Londres, Reino Unido, trabalhando também com inteligência artificial, agora com dados epigenéticos e suas relações com a asma. Esta obra representa não apenas o fruto de uma trajetória acadêmica, mas também a expressão do compromisso do autor em compartilhar conhecimento e inspirar outros a desbravar novos campos de estudo e pesquisa.

PREFÁCIO

Uma certa vez um professor meu de Filosofia da Ciência perguntou à classe se é possível olharmos para o céu e vermos um urubu cor-de-rosa voando. Achei essa pergunta intrigante para um calouro, mas como na maioria das cabeças ainda em fase primitiva de conhecimento científico, a resposta me apareceu rápida e fácil demais: não!

Na verdade, há várias maneiras de se ter um urubu cor-de-rosa voando pelos céus do nosso Brasil. Uma mutação no gene que confere fenotipicamente a cor preta à pena dessa ave pode eventualmente mudar para a cor rosa. Ou, algum infeliz e desalmado sujeito que possa achar interessante pintar um urubu com tinta rosa *pink*. Pode ser um urubu meio albino. Sabe-se lá. A questão por detrás disso aqui é probabilidade desse evento ocorrer, que certamente é pouco provável que aconteça (para a sorte do *Coragyps atratus*, que é mais bonito e imponente do jeito que é).

A bioestatística trabalha muito com essa questão probabilística. Parte de uma ciência muito maior (a matemática), a estatística é responsável por nos mostrar de forma numérica o que está ocorrendo ou até mesmo tentar prever o que possa acontecer em um futuro próximo. Como praticamente todas as ciências necessitam trabalhar com a estatística, por mais básica que seja, resolveu-se empregar o termo bioestatística para a estatística voltada para a área biológica, onde se insere a área da saúde. Matematicamente, não há diferenças, mas sim contextualizando a interpretação de resultados.

Sendo a matemática de exatas, a bioestatística deixa claro que a condição biológica dos seres é tão complexa que é difícil falarmos aqui no rigor da exatidão. Porém, por exemplo, se eu te digo que um paciente que tem um alelo T ao invés de C num determinado gene e isso confere a ele um risco elevado de ter câncer, entenda que eu não estou dizendo que ele terá câncer. Mas, eu sei que você capitou que a probabilidade disso acontecer pode realmente ser alta. Se os cientistas demonstram que a média da altura dos homens é maior do que das mulheres de mesma faixa etária, saiba que terão algumas mulheres mais altas do que alguns homens. Mas, de novo, eu sei que você capitou que na afirmação científica estamos tratando de uma certa generalização.

A bioestatística na verdade é uma ferramenta. De certa forma, o que você aprenderá aqui para aplicar em seus trabalhos acadêmicos e científicos, saiba que você também pode aproveitar para a sua vida, seja no empreendedorismo, nas finanças da casa, ou meramente tentar estabelecer a média de vezes que você corta seu cabelo em um ano e estabelecer quanto gasta por mês com beleza pessoal. O cotidiano é estatístico e probabilístico também.

Uma outra coisa que você precisa saber antes de começarmos nossos estudos é que a intensão deste livro não é o aprofundamento nos cálculos matemáticos, mas sim nos expormos à prática e sabermos interpretar os resultados gerados. Portanto, este é um livro de introdução à bioestatística, e desse modo, os fundamentos matemáticos aqui ensinados é o básico que você precisa conhecer para saber o que está fazendo.

Muitos autores me inspiraram, mas um pequeno trecho dito por Louis Pasteur tem muito sentido na minha motivação a escrever o livro, sobretudo quando fala "... o conhecimento pertence à humanidade.". Estudando por 4 anos a linguagem R, utilizando literaturas fora do meu contexto de uso, me senti na obrigação minimamente cível de sintetizar o básico do R na bioestatística e epidemiologia para que outras pessoas da mesma área possam ser melhor direcionadas a esse universo.

Supondo que você seja um estudante ou um profissional da área biológica (que possivelmente nunca gostou muito de cálculos), espero de coração que você quebre o tabu de que matemática e linguagem da programação são coisas difíceis, para poucos, ou mesmo impossíveis de se aprender e aproveite todo o material aqui disposto.

Bons estudos!

O autor

SUMÁRIO

1 O R E O RSTUDIO.....2

1.1 O QUE É O R?..... 2

1.2 Como instalar o R..... 3

1.3 O que é o RStudio?..... 4

1.4 Como instalar o RStudio..... 4

1.5 Detalhando o RStudio..... 4

1.6 Scripts (criando e salvando linhas de código)..... 9

1.7 Operadores matemáticos e operadores lógicos..... 11

1.8 Objetos..... 12

1.9 Tipos de objetos no R..... 14

1.10 Vetores..... 14

1.10.1 Vetores tipo caractere (*character*)..... 14

1.10.2 Fator (*factor*)..... 15

1.10.3 Inteiro (*integer*)..... 16

1.10.4 Numérico (*numeric*)..... 16

1.10.5 Vetores lógicos..... 17

1.11 Listas..... 18

1.12 Matrizes (*matrix*)..... 19

1.13 Arranjos (*arrays*)..... 19

1.14 Data Frames..... 20

1.15 Funções e seus parâmetros..... 21

1.16 Pacotes e instalação de pacotes..... 23

1.17 Ajuda..... 24

2 DADOS.....28

2.1 Os tipos de variáveis - estatisticamente falando..... 28

2.2 Estrutura dos dados em um banco..... 30

2.3 Mudando vírgula por ponto..... 31

2.4 Construindo um banco de dados do zero..... 32

2.5 Salvando o Banco de Dados..... 36

2.6 Leitura de dados (importação)..... 37

3 MANIPULAÇÃO DE DADOS NO R.....42

3.1 O superpacote tidyverse..... 42

3.2 Vinculando uma pasta de trabalho..... 44

3.3 Observando o banco de dados no R 44
3.4 Ocorrência de dados faltantes 47
3.5 Tratando dados faltantes 48
3.6 Selecionando linhas e colunas 52
3.7 Filtrando o banco de dados 54
3.8 Criando e modificando variáveis 56
3.9 Combinando dados de bancos diferentes (*merge*) 58
3.10 Salvando/Carregando seu ambiente global de trabalho 62
4 AMOSTRAGEM 64
4.1 Amostra probabilística 66
4.2 Amostra semiprobabilística 68
4.2.1 Amostra por conglomerados 68
4.2.2 Amostra sistemática 69
4.2.3 Amostra por quotas 70
4.3 Amostra não probabilística ou de conveniência 70
5 MEDIDAS DE TENDÊNCIA CENTRAL 74
5.1 Média aritmética 74
5.2 Mediana 76
5.3 Moda 77
5.4 Relações entre a média, mediana e a moda 79
6 MEDIDAS DE POSIÇÃO RELATIVA 86
6.1 Percentis 86
6.2 Quartis 87
6.3 Diagrama de caixa ou *boxplot* 88
7 MEDIDAS DE DISPERSÃO 94
7.1 Amplitude 94
7.2 Intervalo Interquartil 96
7.3 Variância e Desvio Padrão 98
7.4 Coeficiente de Variação 101
8 CONCEITOS BÁSICOS DE PROBABILIDADE 106
8.1 Acréscimo/decréscimo percentual sobre um valor 107
8.2 Probabilidade baseada na frequência relativa 110
8.3 Os testes de hipótese e suas probabilidades 111
8.4 O tal do p-valor 118
9 A DISTRIBUIÇÃO NORMAL 122
9.1 Testes da normalidade dos dados 127

9.2 Q-Q plot 131
10 CORRELAÇÃO 134
11 REGRESSÃO 144
11.1 Regressão linear 145
11.1.1 Regressão linear simples 145
11.1.2 Regressão linear múltipla 151
11.1.3 Transformando dados para análises múltiplas 151
11.1.4 Criando modelos de regressão linear múltipla e interpretando seus resultados 154
11.1.5 Comparando modelos de regressão 157
11.1.6 Covariáveis 162
11.1.7 multicolinearidade 163
11.2 Regressão logística 166
12 TESTE DE ASSOCIAÇÃO ENTRE VARIÁVEIS CATEGÓRICAS 172
12.1 Teste qui-quadrado de aderência 172
12.2 Qui-quadrado de independência 174
12.3 Teste EXATO de Fisher 177
13 TESTES DE COMPARAÇÃO DE MÉDIAS 182
13.1 Comparação de uma média amostral com uma média populacional conhecida 185
13.2 Comparação de duas médias de amostragens independentes 188
13.3 Comparação de duas médias de amostras pareadas 194
13.4 Demonstração de resultados para comparação de médias 198
13.5 Comparação de três ou mais médias 200
13.5.1 ANOVA (de amostras independentes) 201
13.5.2 Teste Kruskal-Wallis (amostras independentes) 208
13.6 ANOVA de medidas repetidas (amostras pareadas) 211
13.7 Teste de Friedman (teste não paramétrico para amostras pareadas) 217
14 TESTES DE CONCORDÂNCIA 224
14.1 Kappa de Cohen (concordância de característica) 224
14.2 Bland-Altman (concordância numérica) 227
15 REPRESENTATIVIDADE E CÁLCULO AMOSTRAL 236
15.1 Qui-quadrado 239
15.2 Correlação 242
15.3 Regressão linear simples e múltipla 244
15.4 Regressão logística 249
15.5 Teste t para uma amostra 252
15.6 teste t independente 253

15.7 teste t pareado 255
15.8 ANOVA de uma via 256
15.9 Outros métodos de estimação do n amostral 257
16 TESTE DIAGNÓSTICO 260
16.1 A tabela 2 x 2 260
16.2 Acurácia 263
16.3 Sensibilidade 263
16.4 Especificidade 264
16.5 Valor preditivo Positivo 265
16.6 Valor Preditivo Negativo 265
16.7 Razão de Verossimilhança Positiva 266
16.8 Razão de Verossimilhança Negativa 267
16.9 Prevalência 268
16.10 Tabela diagnóstico na prática R 269
17 A CURVA ROC E O ESTABELECIMENTO DE PONTOS DE CORTE 274
17.1 A curva ROC 274
17.2 Comparando curvas ROC diferentes 283
17.3 Estimando o ponto de corte ideal 285
18 TÓPICOS FUNDAMENTAIS EM EPIDEMIOLOGIA 294
18.1 Acaso, causa e efeito 294
18.2 Prevalência 295
18.3 Incidência 298
18.4 Surto, epidemia, pandemia, endemia 300
18.5 Tipos de Estudos Epidemiológicos 302
18.5.1 Descritivos: Relato de caso e Série de casos 303
18.5.2 Estudo Analítico Ecológico 305
18.5.3 Estudo Analítico Transversal 306
18.5.4 Estudo Analítico Caso-controle 307
18.5.5 Estudo Analítico Coorte 308
18.5.6 Estudos Experimentais 309
18.5.7 Revisão Sistemática 311
18.6 Medidas de associação para estudos retrospectivos 312
18.6.1 Razão de chances (*odds ratio*) 313
18.6.2 Razão de prevalência 315
18.7 Medidas de associação para estudos prospectivos 316
18.8 Trabalhando epidemiologia no R 318

19 APRESENTAÇÃO DOS RESULTADOS 330
19.1 Tabelas..... 330
19.1.1 Tabelas elegantes 333
19.2 Gráficos..... 343
19.2.1 Histogramas..... 346
19.2.2 Gráfico de densidade..... 350
19.2.3 Q-Q plot 352
19.2.4 Gráfico de pontos 356
19.2.5 Gráficos de tendência *smooth*..... 358
19.2.6 Gráficos de linhas 361
19.2.7 Gráficos de área..... 363
19.2.8 Gráfico de barras 365
19.2.9 Diagrama de caixa ou *boxplot* 368
19.2.10 Gráfico de pontos sobrepostos ou *jittered points*..... 371
19.2.11 Gráfico violino 372
19.2.12 Barras de erro..... 375
19.2.13 Adição de textos nos gráficos..... 380
19.2.14 Segmentos de reta 384
19.2.15 Mudando a escala dos eixos gráficos 387
19.2.16 Alterando os títulos dos eixos e legenda..... 391
19.2.17 Alterando o tamanho da fonte dos títulos e das escalas 392
19.2.18 Cores..... 395
19.2.19 Temas 400
19.2.20 Facetas (múltiplos gráficos)..... 404
19.2.21 Múltiplos gráficos de tipos diferentes..... 410
19.2.22 Transformando gráficos do pacote graphics em ggplot..... 417
19.2.23 Sobrepondo gráficos diferentes 420
19.2.24 Mapas 422
19.2.25 Como salvar suas figuras 434
20 REFERÊNCIAS.....438
21 LISTAS 452

Capítulo

1

O R e o RStudio

1.1 O QUE É O R?

O R é uma linguagem e ambiente de programação que também é capaz de gerar gráficos estatísticos. Uma definição interessante dita por outros autores, é mais ou menos assim: encare o R como uma calculadora mais completa e complexa do que uma habitual.

Um dos bons motivos da grande usabilidade do R vem da sua gratuidade, pois faz parte do grande Projeto GNU, que promove ações colaborativas para a geração de software livre. Esse ponto tem uma grande vantagem fora a gratuidade, pois os *bugs* (erros de programação) são rapidamente reparados pela comunidade R em comparação aos softwares pagos, que costumam demorar mais de ocorrer atualizações.

Atualmente, a linguagem R é utilizada em diversas áreas, como geoprocessamento, finanças e análises mais complexas que envolvam a Bolsa de Valores com suas vendas e compras de ações, geração de resultados para pesquisas acadêmicas, ciência de dados que envolve a mineração dos dados de várias fontes, incluindo da internet, para analisar padrões de comportamento, padrões de compra e definir propagandas específicas para um determinado público alvo, enfim. A usabilidade do R é realmente grande e poderosa.

Porém, não é a única linguagem de programa voltada para a estatística e análise de dados. Na verdade, qualquer linguagem de programação pode gerar os resultados estatísticos necessários para respondermos as mais diversas questões. Só para citar outras linguagens interessantes que têm sido utilizadas para esse fim, são: Python, Java e C++. Nesse momento em que vos escrevo, há quase que uma paridade entre a quantidade de cientistas de dados que utilizam R e os que utilizam Python. Muitos deles utilizam ambas linguagens. Qual aquela que você irá escolher? Já te respondo como uma dica: aquela a que você se sentir mais confortável ou aquela à qual o seu nicho profissional mais utiliza.

Fundamentalmente, o ambiente R de programação não é nada atraente à primeira vista (Figura 1.1). Apesar de não trabalharmos diretamente nesse ambiente "seco", ele precisa ser instalado, coisa que aprenderemos fazer já no próximo subtópico. Mas, diante mão, observe que o ambiente R nada mais, nada menos é do que uma janela (ambiente) onde podemos escrever comandos onde o cursor está em vermelho antecedido do sinal ">". Uma coisa interessante que podemos ver nessa ríspida janela é a versão instalada do R e a sua data de lançamento: R version 3.6.3 (2020-02-29), no exemplo.

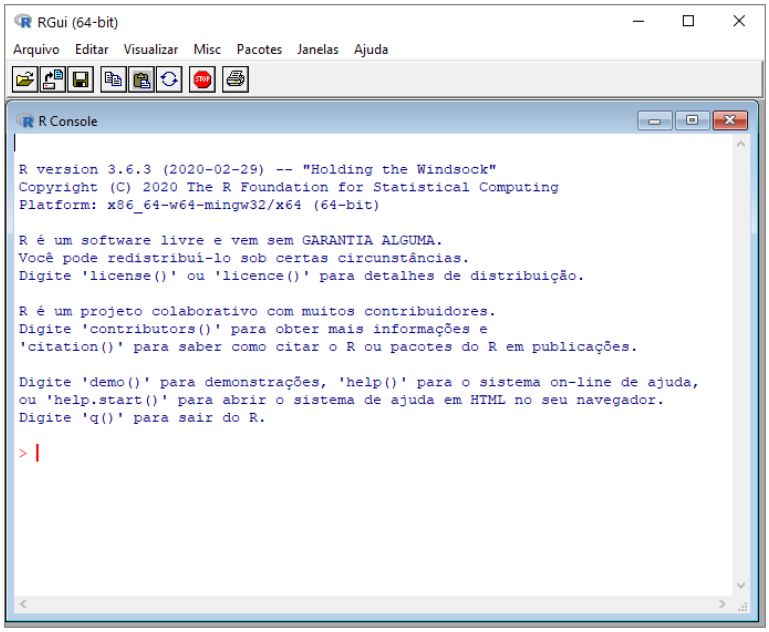


Figura 1.1. O ambiente R de programação.

Estar atento à versão do R é um comportamento interessante, pois os pacotes que o R utiliza vão evoluindo, sendo aprimorados, repaginados e as vezes pode não ser compatível com versões do R mais antigas. Então, uma outra dica inicial é atualizar o R, o RStuido e seus pacotes de tempos em tempos.

1.2 COMO INSTALAR O R

Você, iniciante à programação R, precisa conhecer e desfrutar do site que detém a rede abrangente de arquivos R ou CRAN (*Comprehensive R Archive Network*). Falaremos mais sobre o poder deste site em nossos estudos mais à frente. Por ora, você pode acessá-lo pelo *link*: https://cran.r-project.org. Ainda, você pode fazer uma pesquisa em qualquer site de busca por: CRAN R. Normalmente é o primeiro da lista dentre os sites sugeridos.

Seja para sistemas Linux, Windows ou Mac OS, as instruções para instalação estão na aba *Download and Install R* do CRAN. Na maioria das situações, basta fazer o download do arquivo disponível no site, abri-lo e ir avançando até o final da instalação. Realmente é fácil de se fazer. Quem utiliza Linux, é necessário escolher o arquivo de acordo a distribuição Linux (Distro) que você utiliza, seja Debian, Fedora, RedHat, Suse ou Ubuntu.

1.3 O QUE É O RSTUDIO?

O RStudio em uma IDE (*Integrated Development Environment*), ou seja, um ambiente de desenvolvimento integrado. Sendo uma interface gráfica mais amigável do que o R "seco", é com ele que iremos trabalhar ao longo dessa jornada. Ele é incrível, e você verá o que podemos fazer utilizando essa ferramenta que vem com diversas funcionalidades.

Não quero confundir a sua cabeça, mas quero falar de possibilidades. Entenda desde já que o RStudio não é a única IDE voltada para R. Na realidade, hoje ele suporta inclusive comandos em Python. Porém, há outros ambientes que merecem ser listados: TinnR, Bio7 e Rkward. Fora, ainda, os editores de código mais gerais, como: Eclipse, Notepad++, Komodo edit, Visual Studio, entre muitos outros. Se a curiosidade te inquieta, busque pelos nomes em seu buscador de *sites* de sua preferência e veja algumas imagens da internet para ver se lhe agrada alguns deles ao longo do seu desenvolvimento mais avançado. Por ora, saiba que é muito comum na comunidade que trabalha com o R, falar em R da mesma forma que falamos em RStudio. Ou seja, ele pode falar: "Trabalho com R". Na realidade, ele trabalha no RStudio, usando a linguagem R.

1.4 COMO INSTALAR O RSTUDIO

Primeiro, certifique-se que o R foi previamente instalado. Isso evitará erros desnecessários no futuro. Depois, instale o RStudio, que também é muito simples. Basta acessar o *site* www.rstudio.com e depois clicar em *download*. Lá, você poderá baixar o arquivo instalador do aplicativo, seja para Windows, macOS ou Linux (Ubuntu, Fedora/Red Hat, Debian e OpenSUSE). Após fazer *download*, abra o arquivo baixado e siga as instruções até finalizar a instalação. Normalmente é só clicar em seguir.

1.5 DETALHANDO O RSTUDIO

Perceba na Figura 1.2 que o RStudio é mais que uma só janela. São quatro e cada uma tem uma ou mais funcionalidades.

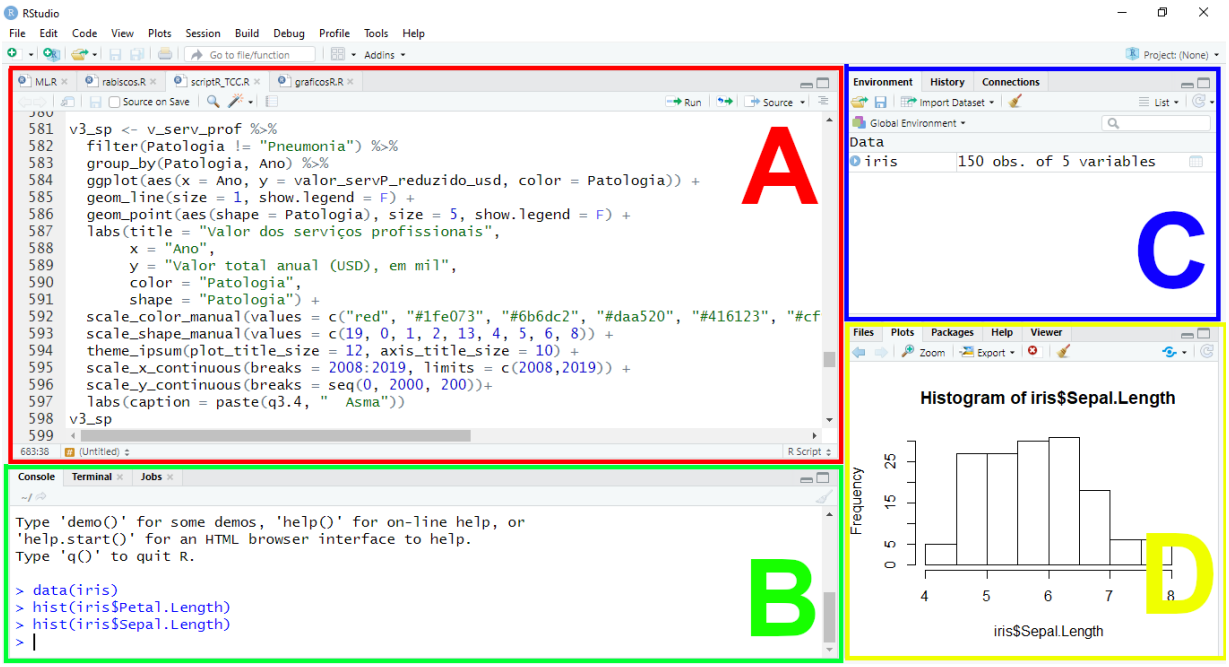


Figura 1.2. O ambiente RStudio.

Vamos detalhar um pouco dada uma dessas janelas e captar o que elas têm de mais importante para a nossa jornada:

A) **Editor**, nossa janela A da Figura 1.2 (e Figura 1.3), será talvez a nossa área mais importante do RStudio, pois será nela que iremos inserir os comandos na linguagem R, solicitar que o R calcule médias, medianas, desvio padrão, enfim. Entenda que o editor de código como um bloco de notas onde podemos rascunhar os comandos que precisamos. A esse rascunho de comandos damos o nome de *script*. Aportuguesando, esse roteiro tem certa semelhança a um roteiro de uma peça de teatro ou de um filme de cinema. Ao desenrolar da história, há a primeira cena, a segunda, a terceira e assim por diante, onde tudo o que os atores devem falar ou ocorrer no palco está descrito no *script*. Levando agora para a nossa realidade, primeiramente precisamos abrir o arquivo que contém os nossos dados, depois modificar possivelmente algumas variáveis, construir outras, fazer as primeiras análises descritivas, depois outras comparativas e por fim gerar um gráfico ou tabela disso tudo. Pois bem, adivinhe onde estará tudo devidamente descrito! No nosso R *script*, que você construirá aqui comigo.

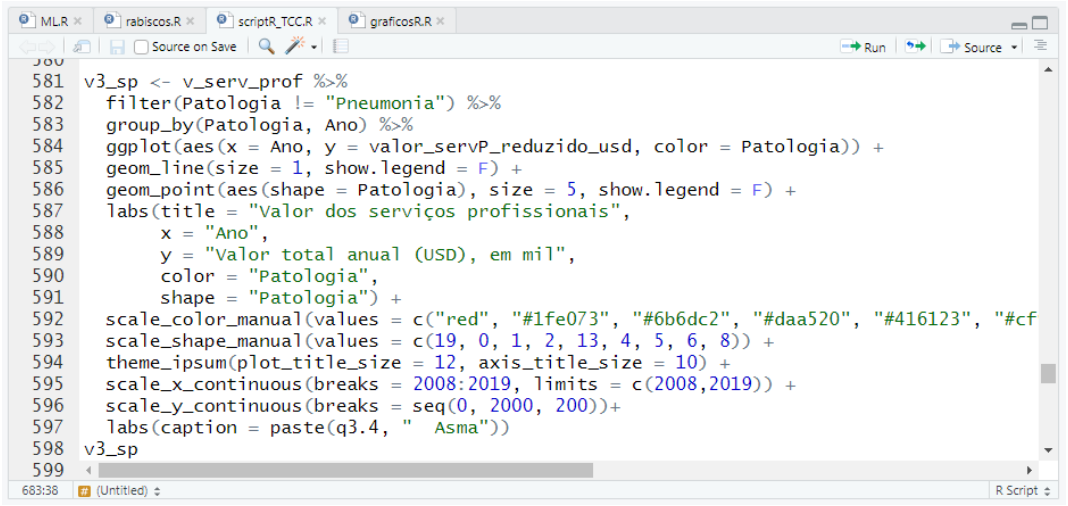


Figura 1.3. Janela do Editor.

B) **Console**, nossa janela B da Figura 1.2 (e Figura 1.4), é aquela "seca" janela do R da Figura 1.1. É uma janela bruta, porém não menos importante, onde podemos observar os comandos que estão ou que já foram rodados. Também, será nela onde visualizaremos boa parte dos nossos resultados. Nela, também poderemos escrever códigos, mas normalmente são comandos mais simples, que necessitamos rodar de uma forma mais objetiva. Se você precisa se lembrar o que foi feito, passo a passo, o console não é das melhores janelas para fazer isso, mas sim o nosso *scritp*.

Observe na Figura 1.4 que foi solicitado (em azul) a abertura de um banco de dados e depois solicitamos a construção de dois histogramas a partir dele. Veja, portanto, que foram três comandos muitos simples e objetivos dados no console. Compare agora esse comando com o da Figura 1.3, que é o código para a construção de um só gráfico. O comando da Figura 1.3 gerará um gráfico mais atraente, contudo, não dá para escrevê-lo adequadamente no Console.

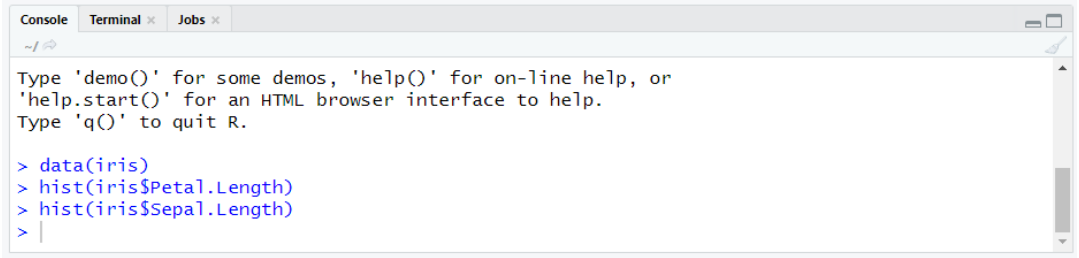


Figura 1.4. Janela do Console.

Vamos fazer um teste? No Console, digite a soma `2 + 5` e posteriormente aperte a tecla "Enter". A resposta foi `[1] 7`? Parabéns! Você está fazendo isso corretamente. O número 1 entre

colchetes significa que o nosso *output* (nosso resultado) tem uma linha só. Então, a partir de agora, divirta-se fazendo qualquer operação matemática simples no console.

Imaginando que você já tenha instalado o R e depois o RStudio, vamos ver qual a versão do R você tem disponível no seu dispositivo. Rode o comando R.Version() no Console. Aparecerá como resultado com muitas linhas, porém a mais importante será a $version.string. Observe abaixo que no momento estou utilizando a versão 3.6.3, lançada no dia 29 de fevereiro de 2020. Até o término deste livro, a minha versão do R já será outra, isso porque a atualização do R, do RStudio e de seus pacotes deve ser uma constância, como dito acima.

```
R.Version()
...
$version.string
[1] "R version 3.6.3 (2020-02-29)"
```

Além de R.Version(), para saber a versão do R, você pode também utilizar o simples comando version. Teste!

C) **Environment**, nossa janela C da Figura 1.2 (e Figura 1.5) é o ambiente onde ficam listados os nossos objetos criados. Mais à frente detalharemos sobre, mas basicamente serão os nossos bancos de dados, fórmulas e funções por nós criadas, algum resultado que a gente salve em um objeto, enfim. Nessa janela há também a aba ***History***, onde estarão listados os comandos que já foram utilizados por você. Isso pode ser feito de forma semelhante apertando a tecla *up* (para cima) do seu teclado lá no Console; na verdade, esse último é um ótimo recurso para melhorar a nossa produtividade no R.

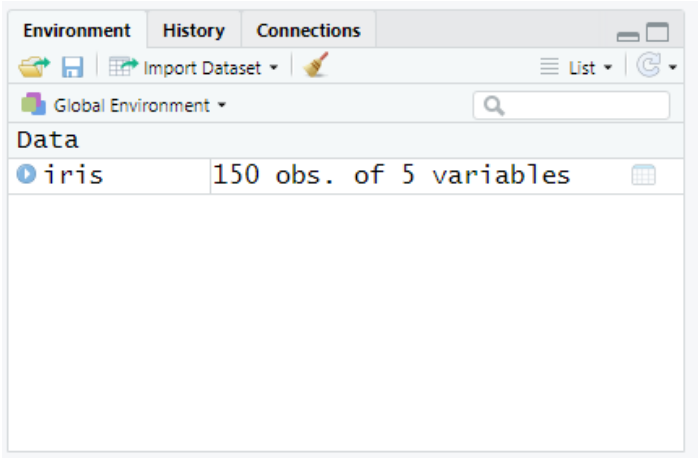


Figura 1.5. Janela com as abas *Environment* e *History*.

Nota: observe o banco de dados chamado *iris* listado em *Environment*. Nele há 150 observações (linhas) e 5 variáveis (colunas).

D) **Files**, **Plots**, **Packages**, **Help** e **Viewer**, fazem parte da nossa janela D da Figura 1.2 (e Figura 1.6). A aba **Files** é um explorador de arquivos. Ela demonstrará pastas e arquivos do seu sistema para serem de fácil acesso durante os seus trabalhos. Ainda nela você poderá criar novas pastas, deletar pastas ou arquivos, renomeá-los e muito mais. É um explorador de arquivos. A aba **Plots** aparecerá os gráficos construídos por você. Nessa janela você poderá dar *zoom* (o que abrirá uma nova janela com o seu gráfico ampliado), exportar esse gráfico transformando-o em um arquivo de imagem ou arquivo *.pdf (apesar de ter outras formas melhores de salvarmos nossos gráficos se o seu rigor é acadêmico, mas essa aba já nos dará uma boa noção de como ficará a imagem). A aba **Packages** é a lista de pacotes instalados no R. Falaremos mais sobre pacotes mais adiante, mas entenda desde já que utilizaremos muitos deles para funções específicas diversas. A aba **Help** é justamente isso que você está pensando: é nosso local para obtermos ajuda; também muito útil em nossa jornada. A aba **Viewer** é um painel de visualização que exibe conteúdo HTML, como aplicações em Shiny, relatórios em RMarkdown (geração de arquivos para Word, PDF ou HTML) ou mesmo visualizações interativas. O escopo deste livro não abrangerá muito as funcionalidades que envolvem a aba **Viewer**, mas pode ser de seu interesse quando você tiver conhecimento um pouco mais avançado sobre o R.

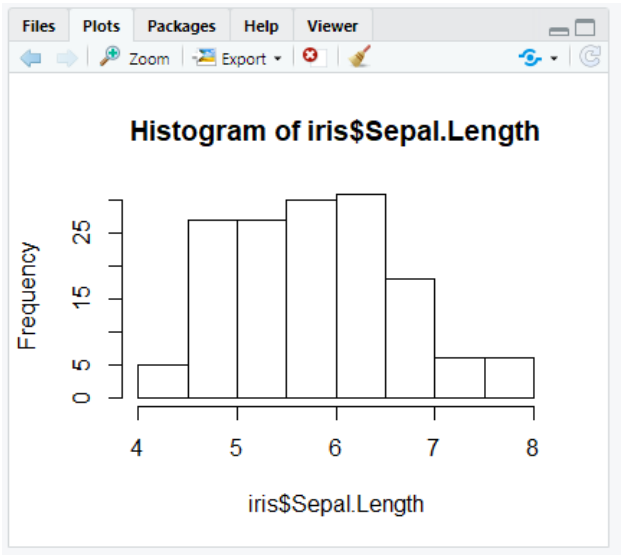


Figura 1.6. Janela com as abas *Files*, *Plots*, *Packages*, *Help* e *Viewer*.

1.6 SCRIPTS (CRIANDO E SALVANDO LINHAS DE CÓDIGO)

Como dito mais acima, o *script* nada mais é do que uma área de criação de códigos e ter isso roteirizado. E, de novo, nada impede que códigos sejam escritos no Console.

Podemos abrir vários *scripts* de uma vez. Para cada trabalho meu, eu tenho um *script* separadamente. Por vezes, esqueço alguma linha de comando que me interesse e vou busca-la em um *script* que fiz de um outro projeto. Vamos exemplificar: para a minha iniciação científica, eu tenho o meu *script* para trabalhar com aquele banco de dados específico, fiz determinadas análises e salvei todo o código usado em um *script* chamado "script_IC.R" Já no mestrado, trabalhei com outro banco de dados, fiz outras análises e está tudo registrado, passo a passo, em um *script* único chamado "script_Msc.R".

Ao longo do nosso curso você verá que essa informação pareça boba, mas quero aqui te mostrar que desde o início é importante ter organização quando se trabalha com linguagem de programação. Se você escreve um só *script* para trabalhos diferentes, facilmente você se perderá entre as linhas de código criadas.

Para criar um *script* novo no RStudio, basta ir no menu File > New File > R script. Ou, se você estiver trabalhando no Windows, basta apertar as teclas de atalho Ctrl + Shift + N. Para salvar um *script*, basta clicar no ícone "Save current", ou ir no menu File > Save, ou usar as teclas de atalho Ctrl + S.

Como dica, crie em seu computador uma pasta para estudos do R. Depois, crie um *script* e salve-o com o nome deste livro ou algo que lembre o livro, algo do tipo: livro_bioestatistica_R. Caso apareça uma janela solicitando o encoding (*Choose Encoding*), dê preferência para o UTF-8. Na pasta em que você salvou o seu *script* verá um arquivo de extensão *.R. Então, se você seguiu a minha dica, haverá o arquivo "livro_bioestatistica_R.R".

Ctrl + Shift + N para criar um novo *script*.
Ctrl + S para salvar o *script* atual.

Se lembra que fizemos a operação aritmética 2 + 5 no Console? Vamos deixar isso um pouco mais desafiador? Peço-lhe que faça a operação: quatro mais cinco, depois diminua esse resultado de três e depois divida o resultado por dois. Para rodar o comando no Editor, o cursor precisa estar na linha do comando desejado e depois você precisa clicar em Run ou só apertar as teclas de atalho Ctrl + Enter (como sugerido acima).

Você pode fazer passo a passo no seu Editor:

```
4 + 5   #Ctrl+Enter ou clique em Run                comando seguido de um comentário
[1] 9                                             Esse resultado você verá no Console

9 - 3   #aperte Ctrl + Enter ou clique em Run
[1] 6

6 / 2   #aperte Ctrl + Enter ou clique em Run
[1] 3
```

Para rodar o comando de uma determinada linha do seu *script*, certifique-se que o cursor esteja nessa linha, então, tecle Ctrl + Enter ou clique em Run.

Outra coisa interessante é o que o caractere *hashtag*, cerquilha ou jogo da velha (#) faz. Na linguagem R ele serve para adicionarmos um comentário logo após ele. Então, na primeira linha, eu solicitei a soma 4 + 5 e fiz um comentário "aperte Ctrl + Enter ou clique em Run".

Hashtag ou cerquilha ou jogo da velha (como queira) serve para pôr comentários ao longo do código. Qualquer coisa que esteja após "#" não será lido pelo R.

Até aí, perfeito. Mas podemos escrever esse conjunto de operações em uma só linha:

```
((4 + 5) - 3) / 2                  #aperte Ctrl + Enter ou clique em Run

[1] 3
```

Saiba que números na linguagem R seguem o padrão Norte Americano, que é diferente do brasileiro quanto às casas decimais e a separação do milhar. O valor 2,5 no R será 2.5. Já o valor 1.500 (mil e quinhentos) será 1,500 ou somente 1500. Isso é uma coisa que você precisa se preocupar desde a criação de um banco de dados. Muitas pessoas trabalham com planilhas Excel e dependendo da versão e demais configurações, todos os números podem seguir o padrão brasileiro separando as casas decimais com vírgula, o que deve ser mudado antes de mais nada. No próximo capítulo terá uma seção a qual aprenderemos algumas formas de mexer com isso.

Ainda na janela do Editor, observe que há um ícone ao lado de *Run*; o *Re-run* (ou Ctrl + Shift + P). Ele serve para rodar a linha que foi anteriormente lida. Mais ao lado, há o ícone do *Source*, que pode ser somente *Source* (Ctrl + Shift + S) ou *Source with Echo* (Ctrl + Shift + Entrer). Clicando sobre ele você estará solicitando que o RStuido rode todos os comandos que estiverem no seu *script*. A diferença é que o primeiro não retornará resultados no console e o segundo sim

os retornará. Peço para você ter um pouco de cuidado com os Sources, pois quando houver um *script* com grande quantidade de linhas de comando, pode demorar a rodar tudo, e poucas vezes é o que a gente realmente necessita.

1.7 OPERADORES MATEMÁTICOS E OPERADORES LÓGICOS

Operadores matemáticos são aqueles os quais podemos realizar as operações mais básicas, como adição, subtração, multiplicação e divisão. Já o uso dos operadores lógicos, também chamados de booleanos, gerará resultados que podem ser verdadeiro (TRUE) ou falso (FALSE). Outras operações matemáticas que não aparecem aqui podem ser realizadas através de funções, que falaremos mais adiante. Por ora, aproveite a lista desses operadores abaixo e teste vários exemplos (Quadro 1.1).

Quadro 1.1. Operadores matemáticos e lógicos do R.

Operadores matemáticos	Descrição	Exemplo	Resultado
+	Adição	5 + 2	7
-	Subtração	5 - 2	3
*	Multiplicação	5 * 2	10
/	Divisão	5 / 2	2.5
:	Sequência	1:4	1 2 3 4
^	Exponencial	5^2	25
%%	Resto da divisão	14 %% 5	4
Operadores lógicos	**Descrição**	**Exemplo**	**Resultado**
==	Igualdade	5 == 5	TRUE
!=	Não igual	5 != 5	FALSE
>	Maior que	9 > 4	TRUE
<	Menor que	5 < 2	FALSE
<=	Menor ou igual	6 <= 6	TRUE
>=	Maior ou igual	1 >= 8	FALSE
&	Operador lógico “E”	x <- 7 < 9 (TRUE) y <- 7 == 9 (FALSE) x & y	FALSE
|	Operador lógico “OU”	x <- 7 < 9 (TRUE) y <- 7 == 9 (FALSE) x | y	TRUE

No caso dos operados lógicos "E" e "OU", muito importantes nas operações condicionais, o R trabalha sempre avaliando o resultado TRUE de duas ou mais expressões. No Quadro 1.1, a operação x&y quer avaliar se x é TRUE e y é TRUE. Como isso não é verdadeiro, o R retorna o valor FALSE. Já na operação x|y deseja-se avaliar se x é TRUE ou y é TRUE. Ou seja, basta x ou y ser TRUE, que o R retornará TRUE.

1.8 OBJETOS

A linguagem R é orientada à objetos, assim como é C++, C#, Java, Python e outras. Mas o que tem a ver isso aí? Bom, vamos imaginar uma casa e esta casa tem os seus objetos, que por sua vez são classificados de acordo a sua funcionalidade. Por exemplo, cadeiras, banquinhos, a poltrona e o sofá são móveis o quais eu posso descansar e normalmente têm quatro pernas. O ventilador e o ar-condicionado têm características diferentes dos objetos anteriores e servem para outra finalidade: refrescar o ambiente. Entenda ainda que todos os objetos presentes na casa ocupam um determinado espaço.

Pois bem, na linguagem R, os objetos criados por nós ocupam espaço na memória do computador que você está trabalhando. Esses objetos podem ser variáveis, matrizes, funções, pode ser um único número, uma única letra, pode ser todo o nosso banco de dados ou mesmo um resultado gerado de uma análise através dele. Então, observe que ao longo do trabalho no R viveremos de criar objetos e analisa-los.

Como exemplo, vamos criar um objeto chamado x que tem o valor 25. Para atribuir valor a um objeto/variável, utilize o símbolo de "menor que" e o símbolo de "menos" juntos (<-). De semelhante forma, você pode utilizar o símbolo de igual (=), mas esse não é bem o padrão do R, mas sim uma funcionalidade que talvez tenha vindo do Python. Digite o primeiro comando abaixo e execute-o. Depois, digite a segunda linha e execute-a.

```
x <- 25
x

[1] 25
```

Pronto! Agora o valor 25 está relacionado ao objeto x que você pode verificar na janela Environment. Podemos fazer diversos cálculos utilizando x.

```
5 * x        #cinco vezes x
[1] 125
```

```
x^2             #x elevado ao quadrado
[1] 625

x - 21              #x menos 21
[1] 4

x / 5                  #x dividido por 5
[1] 5
```

Vamos criar o objeto y e interagir ele com x:

```
y <- 50
y
[1] 50

y – x
[1] 25

y/x
[1] 2

y * x
[1] 1250

#criando um objeto chamado z
z <- (100 - y) / x
z
[1] 2
```

Observe que acabamos de criar um objeto z como resultado de uma operação matemática envolvendo outros objetos. Executando z você verá o resultado no console facilmente.

Sobre as possibilidades de nomes dos objetos, consequentemente das variáveis também, o R não permite que comece com números nem caracteres especiais como: “_”, “,”, “.”, “$”, “#”, “&”, “@”, etc. Fundamentalmente, crie objetos que comecem com letras, podendo ser maiúsculas ou minúsculas.

Para objetos, também evite nomes de funções, constantes ou nomes reservados do R, como: c, q, s, t, C, D, F, I, T, diff, exp, log, min, max, mean, median, sd, pi, range, rank, var, FALSE, Inf, NA, NaN, NULL, TRUE, break, else, for, function, if, in, while. Se você achar fundamental algum desses nomes, aconselho adicionar um número ou uma letra para diferenciar.

Para remover/excluir um objeto basta identifica-lo dentro da função rm(), de *remove*.

```
rm(z)
```

Para remover vários objetos, basta separa-los por vírgula.

```
rm(x, y)
```

Caso queira remover todos os objetos criados de uma só vez, rode o comando abaixo, mas tome muito cuidado com esse código na rotina de análises para não excluir objetos importantes desnecessariamente.

```
rm(list = ls())
```

1.9 TIPOS DE OBJETOS NO R

Voltando à analogia dos objetos de uma casa, cada um deles pertence a uma classe e aqui no R não é diferente. Mas, é importante que você entenda que na estatística também vamos classificar as nossas variáveis. A classificação estatística e a classificação no R de variáveis têm algumas coisas em comum, mas não são exatamente iguais, até mesmo porque o universo da linguagem de computação exige uma maior abrangência nesse sentido. Não se preocupe, pois inicialmente parece ser confuso, mas tem muita coisa intuitiva aqui.

Nesse capítulo trataremos das classes dos objetos/variáveis no R como linguagem de programação. Em outro capítulo mais à frente, falaremos de variáveis na ótica da bioestatística e faremos relações um como o outro.

1.10 VETORES

Vetores são os tipos de objetos mais simples que você possa imaginar e se assemelham mais com os tipos de variáveis estatísticas. Dentre os tipos de vetores eles podem ser do tipo caractere, fator, numérico, lógico ou complexo.

1.10.1 Vetores tipo caractere (*character*)

Esse vetor é feito com as palavras ou frases. Necessariamente esse tipo de variável precisa ser escrita entre aspas (pode ser simples ou duplas: ' ' ou " "). No nosso contexto, podemos criar variáveis que apresentem dados como: “doente” ou “sadio”; “normal” ou “elevado”; “hoje o dia está chuvoso” ou “hoje o dia está ensolarado”. Enfim, vamos criar nossos vetores tipo *character*.

Para saber a classe de um objeto/variável, vamos utilizar a função class() e pôr entre parêntese o objeto ou a variável de interesse.

```
x <- "isso aqui é um vetor tipo character"
class(x)
[1] "character"

y <- 'carro'
class(y)
[1] "character"
```

Podemos ainda criar uma variável com mais de um. Para isso, utilizamos a função para combinar c():

```
z <- c("Estou", "combinando", "muitos", "caracteres")
z
[1] "Estou"      "combinando" "muitos"     "caracteres"

class(z)
[1] "character"
```

Há uma outra maneira mais simples de sabermos a classe de um determinado objeto. Após ele feito, basta encostar o cursor do *mouse* por cima do nome desse objeto lá na janela Environment.

1.10.2 Fator (*factor*)

Fatores são parecidos com caracteres, pois podem armazenar informações na forma de letras e palavras. Porém, nos fatores, intrinsecamente esses caracteres estão divididos por níveis. Então, podem ser até algarismos numéricos, mas eles não terão o valor numérico habitual.

São um tipo de vetor muito importante para a ciência de dados, pois são necessários para muitas modelagens utilizando algoritmos para *Machine Learning* (inteligência artificial), regressão logística, enfim.

```
#vamos criar um vetor tipo caractere chamado f1
f1 <- c('gene', 'DNA', 'mRNA', 'DNA', 'mRNA', 'gene', 'DNA', 'mRNA')
class(f1)
[1] "character"

f1           #observe que é do tipo character
[1] "gene" "DNA"  "mRNA" "DNA"  "mRNA" "gene" "DNA"
[8] "mRNA"

f1 <- factor(f1)      #transformando em fator
class(f1)
[1] "factor"

f1    #observe que agora há levels hierarquizados por ordem alfabética
[1] gene DNA  mRNA DNA  mRNA gene DNA  mRNA
```

```
Levels: DNA gene mRNA
```

1.10.3 Inteiro (*integer*)

Aqui são os números de fato, mas como na matemática temos diferentes tipos numéricos, no R também tempos algumas especificações. Por exemplo, temos os números inteiros como -1, 2, 27, -45, etc. Para especificar se algum objeto ou variável é do tipo inteiro, devemos acrescentar a letra L, em maiúsculo mesmo, para discernirmos dos demais números. Mas não se preocupe especificamente com a letra, pois ela não atrapalhará em nada futuros cálculos. Será registrado na memória apenas o número na forma de inteiro. Desse modo, escreva e observe os tipos de objetos atribuídos abaixo:

```
#número 1 na forma de inteiro
a1 <- 1L
a1
[1] 1
class(a1)
[1] "integer"

a2 <- 37L
a2
[1] 37
class(a2)
[1] "integer"

a3 <- -2L
a3
[1] -2
class(a3)
[1] "integer"

#soma de inteiros
1L + 2L
[1] 3

a1 + a2 + a3
[1] 36
```

1.10.4 Numérico (*numeric*)

Temos também os números reais, os quais comportam números fracionados que apresentam casas decimais, como: 2,5; -1,342; 0,37. A esses, chamamos no R de *numeric*. Esse tipo de vetor numérico é também chamado por uns de *double* (assim como em outras linguagens de programação, como Java e C#). Como os números do tipo *numeric* comportam casas decimais, é muito importante que você lembre que a separação no Brasil é feita por vírgula, mas em boa parte

do mundo é feita por ponto. Desse modo, entenda que para o R o número 2.345 será na verdade dois vírgula trezentos e quarenta e cinco. Vamos a alguns exemplos:

```
#um número contendo três casas decimais
b1 <- 2.251
b1
[1] 2.251
class(b1)
[1] "numeric"

b2 <- 53.7
b2
[1] 53.7
class(b2)
[1] "numeric"
```

Podemos atribuir a forma científica de um número. Umas a letra "e" para informar ao R que é "10 elevado a", ou seja, 10^e.

```
c1 <- 1.23e3              #Esse é o número 1230
[1] 1230
class(c1)
[1] "numeric"

c2 <- 1.23e-3             #Esse é o número 0.00123
[1] 0.00123
class(c2)
[1] "numeric"
```

Podemos escrever c1 e c2 da seguinte maneira também:

```
C1 <- 1.23*10^3
C1
[1] 1230

C2 <- 1.23*10^-3
C2
[1] 0.00123
```

1.10.5 Vetores lógicos

Os vetores lógicos se resumem a TRUE (verdadeiro) ou FALSE (falso).

```
#se 1 foi menor que 2, retornará TRUE
1 < 2
[1] TRUE

#se 1 for maior do que 2, retornará FALSE
d2 <- 1 > 2
d2
[1] FALSE

class(d2)
```

```
[1] "logical"
```

1.11 LISTAS

Aqui, o nome também é bem intuitivo. Um objeto do tipo lista é uma lista de vetores. Podemos criar listas com a função list() e dentro dela poderá ser juntados vetores de tipos diferentes, como você verá nos exemplos abaixo:

```
#criando lista numérica
lista1 <- list(10,20,30,40,50,60,70,80,90)
lista1
[[1]]
[1] 10

[[2]]
[1] 20

[[3]]
[1] 30

[[4]]
[1] 40

[[5]]
[1] 50

[[6]]
[1] 60

[[7]]
[1] 70

[[8]]
[1] 80

[[9]]
[1] 90

#criado uma lista mista
lista2 <- list(c(2,3,4), 7L, 'carro')
lista2
[[1]]
[1] 2 3 4

[[2]]
[1] 7

[[3]]
[1] "carro"
```

A marcação entre colchetes duplos indica a posição do vetor dentro da lista.

1.12 MATRIZES (*MATRIX*)

Muita gente deve lembrar desse nome lá do período do colégio e é justamente a mesma coisa. Aqui, podemos dizer que uma matriz é um conjunto de dados distribuídos entre um determinado número de linhas e colunas. Então, se eu digo que uma matriz é 3 x 2, ela tem 3 linhas e duas colunas, e comportam dentro dela 6 elementos. Portanto, as matrizes são bidimensionais.

A função matrix() nos ajuda a construir diversas matrizes. Vamos ver exemplo:

```
#construindo uma matriz 3x2 seguindo a distribuição dos elementos por linhas
matriz <- matrix(c('a','b','c','d','e','f'), nrow = 3, ncol = 2, byrow = TRUE)

matriz
     [,1] [,2]
[1,] "a"  "b"
[2,] "c"  "d"
[3,] "e"  "f"
```

Quando especificamos “nrow = 3”, dizemos para o R que queremos 3 linhas (*row* em inglês é linha). E “ncol = 2” dizemos ao R que queremos duas colunas. Em “byrow = TRUE” estamos informando ao R que ele distribua os valores da matriz por linha, assim, “b” ficará ao lado de “a”, e não abaixo dele.

1.13 ARRANJOS (*ARRAYS*)

Arrays são objetos que fogem a bidimensionalidade das matrizes. Imagine um armário cheio de gavetas e que em cada gaveta há vários outros armários com suas próprias gavetas contendo objetos variados; é quase isso. Não trabalharemos muito com a construção de *arrays* aqui em nossos estudos, mas certos resultados ficam compartimentalizados de forma semelhante a *arrays*, então é bom ter conhecimento.

```
#construindo um array de três dimensões
a <- array(c('DNA','mRNA', 'enzima'),dim = c(3,3,2))
a
```

Você verá isso no console:

```
, , 1
     [,1]     [,2]     [,3]
[1,] "DNA"    "DNA"    "DNA"
[2,] "mRNA"   "mRNA"   "mRNA"
[3,] "enzima" "enzima" "enzima"

, , 2
```

```
     [,1]     [,2]     [,3]
[1,] "DNA"    "DNA"    "DNA"
[2,] "mRNA"   "mRNA"   "mRNA"
[3,] "enzima" "enzima" "enzima"
```

Perceba que foram criadas duas matrizes semelhantes, mas todas dentro de um objeto só, um *array*.

1.14 DATA FRAMES

Dentre os objetos que foram citados, *data frames* são de longe os mais importantes para nós. Eles são objetos tabulares, ou seja, os quais são tabulados dados de diversas categorias, e isso é o que mais se diferenciam das matrizes. Em uma coluna de um *data frame* podemos abrigar dados tipo caractere, na segunda coluna podem ser dados do tipo fator, na terceira tipo numérico, e por aí vai.

Por conta dessa possibilidade de diversidade entre variáveis é que serão amplamente utilizados por nós como nossos bancos de dados. Ainda, para algumas análises é exigido que o banco esteja no formato de *data frame*, caso contrário podem gerar erros.

Como exemplo, vamos construir à mão um *data frame* no R usando a função data.frame():

```
IMC <- data.frame(genro = c("masculino","feminino","feminino","masculino"),
                  altura = c(152, 171.5, 165, 179),
                  peso = c(81,93, 78,87),
                  idade = c(42,38,26, 27.15)
)

#visualiza no Console
IMC
      genro altura peso idade
1 masculino  152.0   81 42.00
2  feminino  171.5   93 38.00
3  feminino  165.0   78 26.00
4 masculino  179.0   87 27.15

#abre uma aba de visualização de dados no RStudio
View(IMC)
```

Não se assuste se por aí você encontrar um objeto do tipo *tibble*. Vindo do pacote chamado tibble, esse tipo de objeto tem sido usado no lugar dos *data frames*. Na realidade, um *tibble* é praticamente um *data frame*, só que levemente diferente. Um pouco mais grosseiro, mais simplificado, ele força o usuário a tratar certos problemas nos dados já na importação. Eles não mudam o nome das variáveis nesse momento, o que pode acontecer na importação de um *data frame*. Além

do mais, os *tibbles* apresentam um método print() diferenciado dos *data frames*. Mas, em suma, para quem tem os dados limpos, não muda praticamente nada.

1.15 FUNÇÕES E SEUS PARÂMETROS

As funções também podem ser um tipo de objeto. No R, trabalharemos muito com várias delas. Algumas prontas, outras construiremos. Mais de 90% do que faremos no R, executaremos utilizando funções. Com elas, podemos calcular uma raiz quadrada de um numero, somar os valores de toda uma coluna, calcular a média de uma variável, ler um banco de dados, enfim. Elas servem para facilitar a nossa vida e não precisarmos resolver todos os nossos problemas estatísticos do zero.

Vamos utilizar uma função para calcular a raiz quadrada de 21, a função chamada sqrt(), abreviação do inglês *square root*.

```
sqrt(21)
[1] 4.582576
```

Podemos guardar esse resultado em um objeto que chamaremos de raiz_quadrada e depois rodaremos ele para observamos o resultado que foi guardado.

```
raiz_quadrada <- sqrt(21)
raiz_quadrada
[1] 4.582576
```

Agora, vamos reduzir a número de casas decimais para duas após a vírgula utilizando uma outra função, a round().

```
round(raiz_quadrada, 2)
[1] 4.58
```

Para ficar mais coerente e não muito trabalhoso, poderíamos guardar esse valor arredondado no objeto inicial, da seguinte forma:

```
raiz_quadrada <- round(sqrt(21), 2)
raiz_quadrada
[1] 4.58
```

Perceba que é possível utilizar funções dentro de outras funções. Observe também que esse objeto e o seu valor está guardado em *Environment*, que consequentemente estão na memória do seu computador.

Vamos a outro exemplo. Com a função seq() podemos criar uma sequência de números. Teste o comando:

```
seq(1, 9, 2)
[1] 1 3 5 7 9
```

Pois bem. Acabamos de gerar uma sequência que começa no 1, termina no 9 e ela deve servir pulando 2 em 2. Na realidade, o que tem por detrás dessa função são os seus parâmetros: "from", "to" e "by". Os valores usados (1, 9 e 2) são os seus argumentos.

No caso acima, eu não explicitei os parâmetros por já conhecer a sua ordem na sintaxe da função, mas podemos especificá-los, como mostrado abaixo. Porém, quando eu especifico um argumento de um determinado parâmetro dentro de uma função, não preciso rigorosamente pôr na ordem padrão da sintaxe da função. Veja as possibilidades de geramos a mesma sequência usando a mesma função, mas em ordem de atribuição diferentes:

```
seq(1, 9, 2)
[1] 1 3 5 7 9

seq(from = 1, to = 9, by = 2)
[1] 1 3 5 7 9

seq(to = 9, from = 1, by = 2)
[1] 1 3 5 7 9

seq(by = 2, to = 9, from = 1)
[1] 1 3 5 7 9
```

Quando utilizamos a função round(), usada mais acima, especificamos os dois dígitos após a vírgula de um número através do parâmetro chamado "digits" da função. Dessa forma, poderíamos reescrever o comando mais acima da seguinte maneira:

```
raiz_quadrada <- round(sqrt(21), digits = 2)
```

Às vezes é importante que digitemos o parâmetro, mas não será a maioria das ocasiões. Costumeiramente saberemos a ordem dos parâmetros de cada função e assim não precisaremos escrever tanto código para realizar a mesma tarefa. Ou seja, otimizaremos nosso tempo.

Por fim, podemos criar as nossas próprias funções, utilizando justamente uma função chamada function(). Quando criamos uma função qualquer, ela fica guardada na memória, como um objeto global. Exemplificando, vamos criar uma função que some duas unidades a qualquer número que a ela for designado. Chamaremos essa função de "somar_dois". O "x" será o único parâmetro dessa função, que o usuário deve pôr o valor que ele quer que se some a 2.

```
#criando função que some duas unidades ao número cedido
somar_dois <- function(x){
      x + 2
}
```

Observe que ela está registrada em Environment. Então, vamos testar ela com os números 4, o 25 e o 2.537.

```
somar_dois(4)
[1] 6

somar_dois(25)
[1] 27

somar_dois(2537)
[1] 2539
```

Perceba que eu não precisei escrever o nome do parâmetro dentro da função, mas poderia:

```
somar_dois(x = 4)
[1] 6
```

1.16 PACOTES E INSTALAÇÃO DE PACOTES

Pacotes nada mais são do que um conjunto de funções que podemos utilizar para as mais diversas tarefas. O R já vem com alguns pacotes nativamente, como o *R base*. Esse você não precisará instalar, porém, há muitas funções que esse pacote básico do R não tem e que necessitaremos.

Se você entrar no endereço do CRAN (https://cran.r-project.org), indo no menu *Packages*, clicando em *Table of available packages, sorted by name*, você terá uma dimensão da riqueza de pacotes que o R nos oferece oficialmente. E ainda não são todos. Há a possibilidade de instalação de outros por fora. Mas, categorizando, há pacotes voltados para ensaios clínicos, genética, finanças, inteligência artificial, psicometrias, ciências sociais e tantos outros campos da ciência de dados.

Para instalar um pacote no R, precisamos utilizar a função install.packages(). O nome do pacote que se deseja instalar deve estar obrigatoriamente entre aspas simples ou entre aspas duplas. Como exemplo, vamos instalar um pacote que é importantíssimo para a geração de gráficos, o ggplot2. Digite e execute o comando abaixo:

```
install.packages("ggplot2")
```

Você deve ver muitas sentenças sendo lançadas no console e pequenas janelas irão abrir e fechar rapidamente. Depois de instalado, cada vez que necessitarmos utilizar funções do nosso novo pacote, precisaremos “chamar” ele antes de mais nada, utilizando a função library(). Para

abrir o pacote instalado, não é obrigatório pôr o seu nome entre aspas. Então, digite e execute o seguinte comando:

```
library(ggplot2)
```

Nada de muito especial ocorrerá, mas o nosso pacote já estará disponível na memória para o nosso uso. Então, entenda a partir de agora, que sempre que formos utilizar alguma função fora dos pacotes nativos do R precisaremos carregar a biblioteca do pacote previamente com library().

1.17 AJUDA

Uma grande vantagem de utilizarmos o R e o RStudio é porque nunca estaremos sozinhos. A comunidade R é muito grande e bem colaborativa. Há muito material que você pode utilizar como aprendizado. Seja outros livros, seja postagens em *blogs*, postagens no Instagram, grupos no Facebook, vídeos do YouTube. Buscando, você achará muita coisa. Porém, aqui, vamos falar da ajuda oficial do R.

Entenda que cada pacote que você for utilizar no R tem uma documentação relacionada. Isso é ótimo, pois nada mais é do que os manuais necessários para aplicarmos as funções que necessitamos. Então, todo pacote do R terá um manual de ajuda.

O primeiro caminho para acessar uma documentação será no próprio CRAN do R. Entre no site https://cran.r-project.org, acesse o menu *Packages*, depois *Table of available packages, sorted by name* e escolha qualquer um dos pacotes listados. Após clicar no pacote escolhido, procure por "Reference manual:" e clique no link que dá acesso ao arquivo PDF de mesmo nome do pacote logo ao lado. Nele, você verá a descrição de todas as funções que tem esse pacote, inclusive exemplos de como utilizar cada uma delas, o que é muito importante.

O segundo modo de ajuda é a aba Help da janela D da Figura 1.2. Veja melhor essa aba na Figura 1.7, onde estão três partes lado a lado da documentação da função round() que utilizamos mais acima para arredondamento de um número. Observe inicialmente que a documentação mostra o nome da função e a qual pacote ela pertence (pacote: {base}). Fala explicitamente para que ela serve: *Rounding of Numbers*. Estão presentes os argumentos que podemos modificar, a sequência padrão da sintaxe desses argumentos, e, dentre outras coisas, exemplos de usos da função.

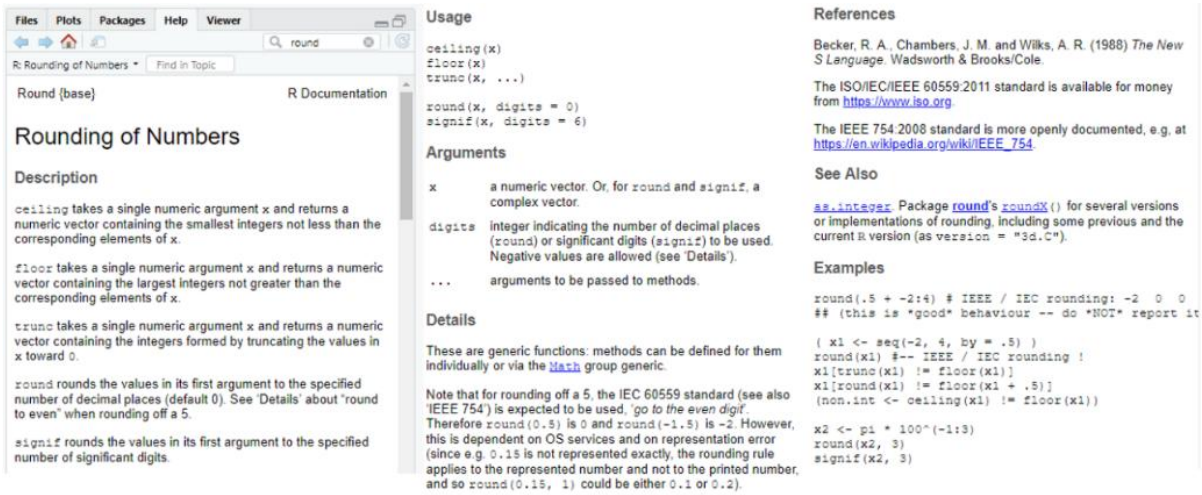


Figura 1.7. Aba Help e a busca de ajuda da função round().

Ter essa ajuda no próprio RStudio é uma "mão na roda". Mas para ela estar ativa, você precisa obrigatoriamente estar com acesso à internet. Um caminho rápido para acessar essa ajuda é digitar o nome da função e com o cursor em cima dela aperta a tecla de atalho F1. Podemos por uma interrogação antes do nome da função de rodar seu código.

```
?round()
```

Teclas de atalho são importantes para maximizar a nossa produtividade. Para ver todas elas, no menu principal, clique em Help > keyboard Shortcuts Help ou tecle Alt + Shift + K.

Imagino que, como foi comigo, nesse início de aprendizado você tenha se assustado um pouco com a grande quantidade de funções que o R apresenta. Parece que temos que saber de todas elas, mas isso não é verdade. Porém, uma coisa é certa, precisamos conhecer muitas delas. Um outro processo que serve para nos ajudar nessa jornada é darmos uma olhada nos *Cheatsheets* do RStudio. *Cheatsheets* são resumos dos principais pacotes utilizados pela comunidade R. São quase que mapas mentais desses pacotes.

Nesse momento, vamos dar uma olhada no *Cheatsheets* do próprio RStudio. No menu *Help*, vá em *Cheatsheets* > *RStudio IDE Cheat Sheet*. Será baixado no seu computador um arquivo PDF. Abra o arquivo, reveja pontos aqui já discutidos e conheça outros tantos. Para visualizar a lista completa dos *Cheatsheets* disponíveis, vá em *Help* > *Cheatsheets* > *Browse Cheatsheets*, ou acesse https://rstudio.com/resources/cheatsheets/.

Dicas de *Cheatsheets* que serão interessantes ao longo desse nosso progresso:

- Básico do R: *Base R*;
- Importação de dados: *Data Import Cheatsheet*;
- Manipulação de dados: *Data transformation with dplyr* (existe versão em português);
- Construção de gráficos: *Data visualization with ggplot2* (existe versão em português).

Outro método de buscar ajuda em relação a uma determinada função ou mesmo tarefa é recorrer à inteligência artificia (IA). Atualmente, o ChatGPT (chat.openai.com), ChatSonic (app.writesonic.com) e o Gemini (gemini.google.com/app) são os modelos treinados que têm gerado melhores resultados nessas buscas por ajuda com códigos. Certamente, no momento em que você estiver lendo este livro já terão várias outras IAs para fazer isso, pois percebo a crescente criação dessas ferramentas por diversas empresas de tecnologia.

Para buscar ajuda através de IAs é importante saber dar entrada nos *prompts* de comando da melhor forma possível para que a IA "entenda" o que você realmente necessita. Um exemplo de *prompt* que pode ser usado para nos ajudar a remover objetos que comecem com a letra "T" pode ser o seguinte: na linguagem R, como remover objetos que comecem com a letra T?

Perceba três pontos nesse simples *prompt*: (1) ele chama a conversa para o tema específico: linguagem de programação R; (2) simples e objetivo; (3) as palavras-chave para o que queremos estão lá (remover, objeto, começar, letra, T). Caso você não esteja obtendo bons resultados nesse tipo de busca, tente achar sinônimos das palavras-chave ou traduza o *prompt* para o inglês. A tradução pode ser interessante, pois essas inteligências artificiais são treinadas com textos em inglês em sua maioria.

Capítulo

2

Dados

Dados são registros capturados de uma população. Se coletarmos as alturas das crianças da 5ª série de uma determinada escola, cada valor será um dado. Já a altura será a nossa variável, que é o conjunto desses dados específicos de altura.

Se um pesquisador realiza um questionário para avaliar a satisfação do atendimento de um laboratório de análises clínicas, cada informação que um certo paciente der será um dado. Cada pergunta do questionário gerará uma variável diferente. Nada impede que criemos variáveis novas a partir de variáveis já pré-existentes. Faremos isso ao longo da nossa jornada, mas é interessante que você tenha em mente que podemos ajustar dados brutos de acordo com a nossa conveniência para encontrar a verdade por detrás dos números.

2.1 OS TIPOS DE VARIÁVEIS – ESTATISTICAMENTE FALANDO

Falamos no capítulo passado sobre os tipos de variáveis para a linguagem R. Sinto muito se por ventura eu venha a confundir um pouco a sua mente, mas para a estatística a classificação de tipos de dados é um pouco diferente do que é no R (ou em qualquer outra linguagem de programação). Saber classificar estatisticamente as variáveis é muito importante, porque isso recai nas futuras análises que faremos.

Primeiramente, variáveis, mais do que uma coluna em um *data frame*, é uma característica de interesse para uma pesquisa. Digamos que um biólogo queira estudar o comportamento de saguis (*Callithrix jacchus*) que vivem na mata e saguis que vivem em zoológico. Em sua prancheta, ele vai a campo e começa a anotar as características de interesse para todos os saguis que encontrar nos dois diferentes ambientes. Cor prevalente da pelagem (qualidade), tempo de horas em atividade (número), quantidade de trocas de galho (número), apresentar agressividade (qualidade), sexo (qualidade), idade estimada (número), e por aí vai. Todos esses atributos são variáveis que futuramente serão avaliados estatisticamente de formas completamente diferentes.

Curiosamente, a depender da área que se esteja trabalhando, como *Machine Learning*, por exemplo, não é tão comum chamar em inglês variáveis de *variables*, mas sim de *features*. Entretanto, é exatamente a mesma coisa. Então, áreas diferentes podem ter nomes diferentes para a mesma coisa.

Como visto acima, há variáveis que expressam um valor, um número, uma quantidade. São, portanto, variáveis quantitativas. Só que ainda há certa diferença entre os números, por

exemplo, podemos dizer que estavam na rua 5 indivíduos. Perceba não existe o sentido de existir 5,5 (cinco e meio) indivíduos. Dessa forma, às variáveis quantitativas que expressam sentido de número inteiro, chamamos de **variável quantitativa discreta**. Exemplos: a quantidade de filhos de uma pessoa, número de alunos em uma sala de aula, número de vezes que uma pessoa foi atendida em um serviço de saúde, etc. Na linguagem R, seria equivalente a *interger*.

Quando formos para um laboratório de análises clínicas e solicitarmos uma dosagem de glicose em jejum, no laudo o resultado pode vir o valor 87,54 mg/dL, por exemplo. As variáveis numéricas que podem expressar casas decimais, melhorando a sua precisam, portanto, são chamadas de **variáveis quantitativas contínuas**. Exemplos: o valor gasto por um serviço, a altura das pessoas, o peso, o Índice de Massa Corpórea (IMC), o comprimento de um determinado animal, etc. Observe que todas essas variáveis podem apresentar vários números após a vírgula e isso combina com o tipo *numeric* lá no R.

As variáveis que expressam qualidade, são as qualitativas. Por exemplo, o sexo de uma pessoa pode ser masculino ou feminino; uma variável que signifique o desfecho de estar doente ou não (sim ou não); a cor da pele; a cor dos olhos; o país de origem, etc. Observe que os dados aqui postos são equivalentes à *character* ou *factor* no R. Podem apresentar duas ou mais classes: sim ou não; país de origem (Brasil, Chile, Uruguai, Argentina, Peru, Bolívia), respectivamente. Essas são as **variáveis qualitativas nominais**.

Veja que o tipo de variável logo acima não assume uma ordem. Para diferenciar isso, temos as **variáveis qualitativas ordinais**. Exemplo de variável qualitativa ordinal é a ordem de chegada de atletas participantes de uma maratona: primeiro, segundo, terceiro, quarto. Pode ser a hierarquia dentro de uma empresa: chefe, gerente de departamento, secretária, auxiliar administrativo. Pode ser o grau de tempo de estudo dentro da academia científica: graduação, mestrado, doutorado, pós-doutorado.

Para resumir e fecharmos esse tópico com segurança, observe a Figura 2.1 com um sumário desses quatro tipos de variáveis estatisticamente falando.

As variáveis qualitativas também podem ser chamadas de variáveis categóricas (remete a categorias), enquanto as quantitativas podem ser chamadas basicamente de numéricas.

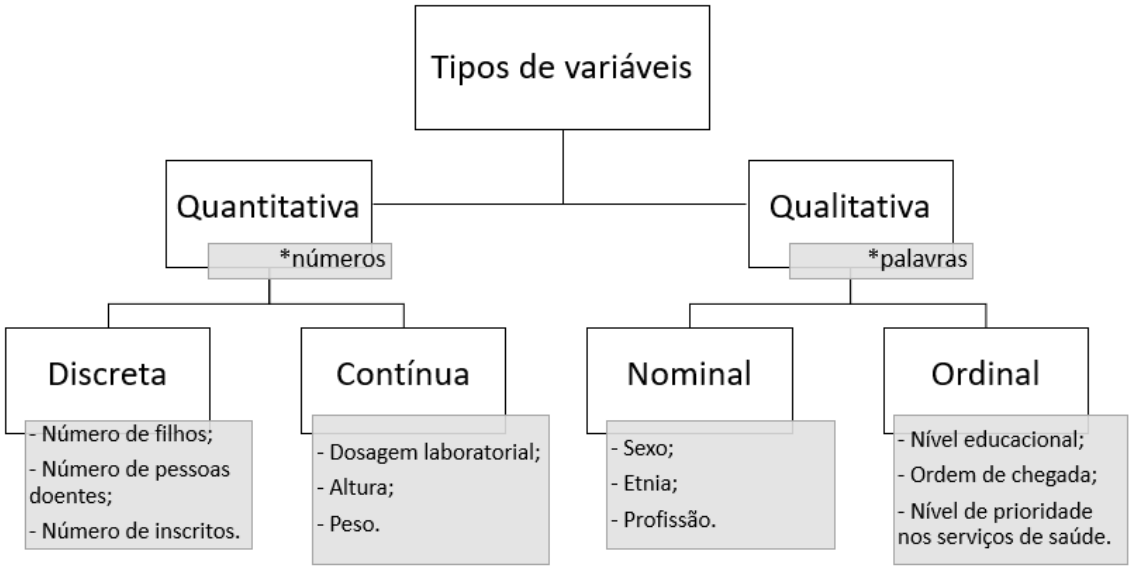


Figura 2.1. Os quatro tipos de variáveis na estatística.

2.2 ESTRUTURA DOS DADOS EM UM BANCO

Sendo dado um registro, qualquer informação pode ser um dado. Um conjunto de dados pode ser um banco de dados, como este livro o é. Aqui você vê várias letras, diversos números, várias imagens. Vídeos, por exemplo, também são dados, pois tem sequência de imagens, textos, enfim. Contudo, esses dados não estão estruturados. Dados não estruturados podem ser imagens, vídeos, áudios, textos de um livro ou de um site/blog/rede social, poque todos esses exemplos há informação. E te digo uma coisa, há muita informação. Entretanto, não há um padrão conformacional confortável para trabalhar estatisticamente com eles. Inclusive, o trabalho mais árduo dos cientistas de dados é justamente modelar os dados para serem analisados.

Dessa forma, trabalharemos na bioestatística com dados estruturados. Para termos um banco de dados estruturado, precisamos preencher as colunas com as variáveis e as linhas desse banco com cada observação. Entenda observação aqui como sendo um paciente, um animal de laboratório, uma planta que serviu como amostra, uma colônia de bactérias, enfim. Dê uma olhada na Figura 2.2 de um banco de dados hipotético feito em Microsoft Excel. Note que cada linha se trata dos dados referentes a um só indivíduo, enquanto que cada coluna é uma característica dele.

	A	B	C	D	E	F
1	**Nome**	**Idade**	**Sexo**	**Altura (m)**	**Peso (kg)**	**IMC**
2	João	51	Masculino	1.75	79.6	25.9918
3	Sandra	46	Feminino	1.54	58.3	24.5826
4	Maria	57	Feminino	1.63	84.4	31.7663
5	Luciano	37	Masculino	1.78	90.1	28.4371
6	Madalena	32	Feminino	1.61	72.3	27.8924
7	Ingrid	19	Feminino	1.77	61.5	19.6304
8	Ricardo	21	Masculino	1.82	75.3	22.7328
9	Raisa	27	Feminino	1.59	59.3	23.4564
10	Pedro	30	Masculino	1.83	99.8	29.8008

Figura 2.2. Banco de dados estruturado.

2.3 MUDANDO VÍRGULA POR PONTO

Como foi dito no capítulo anterior, o R divide as casas decimais de um número contínuo seguindo o padrão norte-americano, ou seja, utilizando ponto. Observe na Figura 2.2 que os valores numéricos estão dessa maneira, mas não é sempre que o Excel reconhece os números dessa forma. A depender da versão que você esteja utilizando, o Excel pode reconhecer o padrão atribuído a sua língua nativa e deixar a forma numérica aportuguesada.

Isso gera muito problema para quem trabalha com outras plataformas de análises estatísticas. Acredite, pode ser o gerador de muita dor de cabeça. Dessa forma, precisamos nos preocupar com esse fato desde já e tentar reverter isso caso seja necessário.

Se você tiver um banco de dados em Excel e os dados numéricos estiverem suas casas decimais separadas por vírgula, selecione a coluna que deseja modificar, depois, na aba *Página Principal* vá no botão *Localizar e Selecionar* e selecione *Substituir*. Na janela que abrir, coloque em *Localizar* a "," e em *Substituir por* o ".". Logo após, clique no botão *Substituir tudo* e com a janela que abrir, certifique-se da quantidade de células que serão modificadas e clique em *Ok*. Esse passo a passo soluciona o problema na maioria das vezes.

Como trabalharemos com outros tipos de arquivos, como a extensão TXT e CSV, abrindo o Bloco de Notas do Windows você consegue fazer a mesma substituição de forma semelhante. Mas tome cuidado, pois no Bloco de Notas do Windows não há a possibilidade de selecionar colunas, então, a substituição ocorrerá por todo o arquivo.

A melhor forma de não passar por esse tipo de problema é construir o banco já no formato aceito como padrão pelo R, que certamente servirá para muitas outras plataformas analíticas. Então, a minha dica é simples: **não utilize vírgula, mas sim ponto** para separar decimais.

Caso você já tenha um banco de dados em mãos onde os números estão com vírgula, há sim um meio de reverter isso lá no próprio R. Então, não precisa se desesperar. Veremos como proceder mais à frente em leitura de dados. A conversa nesse momento é: ter cuidado com essa diferença.

2.4 CONSTRUINDO UM BANCO DE DADOS DO ZERO

Uma coisa precisamos ter em mente, se você pretende se inteirar sobre bioinformática, é: você não trabalhará com um só tipo de arquivo de banco de dados. Atualmente há diversos tipos de arquivos e a seguir lhe trarei os mais utilizados, que podem ser transformados para outros tipos de extensões para serem lidos por programas específicos de bioinformática.

O mais comum na verdade é fazer com que o R leia arquivos de tipos distintos vindos de outras plataformas. No meio da bioinformática, mesmo que as extensões dos arquivos sejam talvez incomuns, boa parte desses arquivos podem ser lidos pelo Bloco de Notas do Windows ou programas similares, cujo *encoding* normalmente o UTF-8. Fora o bom e velho Bloco de Notas, posso citar o Notepad++, que pode ser útil similarmente, é gratuito e distribuído para diversos sistemas operacionais.

O Excel, ou qualquer outro programa *office* relacionado (por exemplo: LibreOffice, WPS Office, OpenOffice, Google Docs, entre outros), pode ser um bom início de construção de um banco de dados. Isso se deve à facilidade visual quanto a estruturação do banco. Lembrando que é ideal respeitar colunas como sendo variáveis e linhas como sendo cada observação, no Excel podemos ir preenchendo as células de acordo com a demanda de dados gerados.

Vamos seguir com um exemplo utilizando a Figura 2.2. O interesse do pesquisador é avaliar o índice de massa corpórea dos profissionais da saúde que estão na linha de frente em combate a uma epidemia tal. No primeiro dia de captação de dados, o pesquisador vai a campo levando a sua prancheta e caneta para tomar nota do seu objeto de estudo. Nesse primeiro dia ele consegue informações apenas do João. Chegando em seu escritório, o pesquisador começa a tabular os dados no Excel. Primeiro ele dá os nomes às variáveis e depois começa a completar as informações colhidas de João. O pesquisador salva o arquivo e no outro dia ele consegue captar os indivíduos:

Sandra, Maria e Luciano. Faz a tabulação desses dados e salva o arquivo. Ao final da captação de dados, ele pode ter tabulado 2 mil profissionais de saúde e seu n amostral talvez esteja satisfatório e ele para por aí.

É importante salientarmos algumas coisas importantes sobre essa tabulação, mesmo que hipotética: (a) dados faltantes podem ocorrer, mas devemos lutar para evita-los; (b) para trabalhar com seres humanos e animais, é necessário que a pesquisa passe por Comitê de Ética (sem a aprovação, não é feita a coleta de dados); (c) o pesquisador precisa desse banco de dados muito bem guardado, tanto no cuidado referente a queima do HD do seu computador e perder essas informações, quanto o cuidado de extravio dessas informações para *hackers* ou outras pessoas que não fazem parte da pesquisa (Lei Geral de Proteção de Dados Pessoais - LGPD).

Dentro do processo de tabulação de dados, salve arquivos diferentes com nomes diferentes que tenha uma sequência lógica de quando foi feito. Exemplo: banco1(data), banco2(data), banco3(data), etc. Se um arquivo corromper, você terá menos trabalho na "retabulação".

Para diminuir o risco de perda do seu banco de dados, salve-o em serviços da nuvem, como Google Drive, Dropbox, entre outros. Caso o HD do seu computador falhe por qualquer motivo, seus arquivos estarão salvos e em fácil acesso.

Muitos podem achar que o R seja uma plataforma "seca" visualmente, mas não é bem verdade. Os seus desenvolvedores nos permitiram criar banco de dados de diversas formas, inclusive visualizando células como assim é no Excel. No RStudio é chamado de Editor de Dados.

Primeiramente, precisamos criar um objeto que seja do tipo *data frame*. Depois, modificaremos esse objeto utilizando o editor de dados com a função edit(). Experimente escrever e rodar os comandos abaixo.

```
#construindo um objeto tipo data frame com 0 variáveis e 0 observações.
banco <- data.frame()

#tabulando e salvando os dados no objeto acima criado
banco <- edit(banco)
```

O que ocorrerá depois do segundo comando é a abertura da janela do Editor de dados, como na Figura 2.3. Nessa janela você pode inserir qualquer tipo de dado, de forma semelhante à inserção de dados em Excel. A Figura 2.4 é um exemplo desse tipo de inserção e construção do nosso

banco. Clicando em cima do nome das variáveis podemos trocar o nome delas e estabelecer o tipo de variável que queremos, se *numeric* ou *character*. No caso da Figura 2.4 estamos modificando a variável da segunda coluna: var2. Quando terminar de digitar os dados, basta fechar a janela e os dados serão salvos no objeto "banco", como denominamos acima.

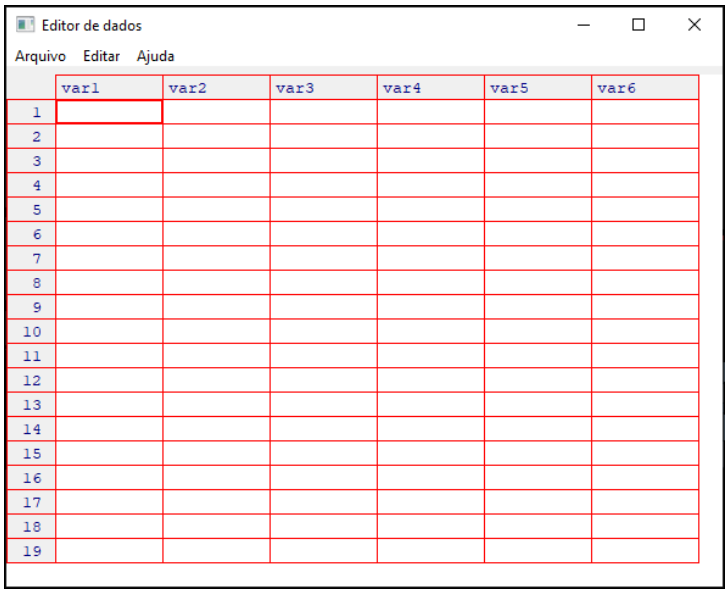


Figura 2.3. Janela do Editor de dados no RStudio.

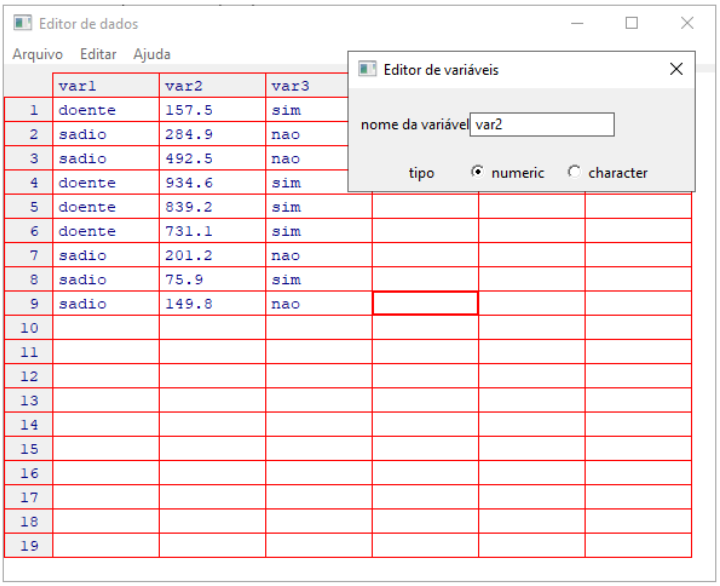


Figura 2.4. Inserindo dados e modificando o nome e o tipo de variáveis.

Há várias outras maneiras de se construir bancos de dados no R. Para a construção de um banco pequeno, por exemplo, podemos começar atribuindo separadamente cada variável e depois juntar todas elas em um *data frame*. Para isso, precisamos da função chamada data.frame(). Veja:

```
#construindo as variáveis
animal <- c('cao', 'gato', 'gato', 'cao', 'cao')
diagnostico <- c('cancer', 'dermatite', 'cancer', 'nefriteA', 'parvovirose')
consultas <- c(9, 2, 13, 6, 3)
progressao <- c('cura', 'cura', 'obito', 'obito', 'cura')

#formando o banco em data frame
banco <- data.frame(animal, diagnostico, consultas, progressao)

#vizualizando o banco de dados
banco
  animal diagnostico consultas progressao
1    cao      cancer         9       cura
2   gato   dermatite         2       cura
3   gato      cancer        13      obito
4    cao    nefriteA         6      obito
5    cao parvovirose         3       cura
```

Dicas de como nomear variáveis/objetos já foram dadas anteriormente, mas nessa seção gostaria dar uma dica extra: evite separar nomes compostos de variáveis com o espaço. Dê preferência ao caractere "_" ou inicie a segunda palavra com letra maiúscula. Exemplo: data_de_nascimento ou dataDeNascimento. Se possível ainda, evite nomes compostos ou grandes demais.

Por fim, para quem for trabalhar com dados de pessoas, será preciso se preocupar com o sigilo. Só quem tem acesso às informações sigilosas, que possa identificar o participante da pesquisa de fato, é o pesquisador responsável e/ou uma pessoa de sua extrema-confiança. Somente eles deverão ter acesso ao nome completo da pessoa, cadastro de pessoa física (CPF), registro geral (RG) e outras informações sensíveis. Se você é pesquisador, dê preferência em pôr as iniciais dos nomes dos participantes para a variável "nome", por exemplo. Então, Luciano Gama da Silva Gomes pode virar um registro tipo LGSG. Podem ocorrer repetições das siglas e para evitar isso você pode correlacionar com a data de nascimento ou ainda - o que eu mais recomendaria - criar um número de registro para cada indivíduo da pesquisa, estabelecida como uma variável categórica.

Caso os dados precisem ser enviados para outro pesquisador, para fazer uma meta-análise, por exemplo, o sigilo dos participantes estará salvaguardado assim que removidos do bando a ser entregue as variáveis sensíveis.

2.5 SALVANDO O BANCO DE DADOS

Podemos salvar nosso banco de dados feito no RStudio utilizando a função write.table(). Na realidade, o que faremos é salvar um objeto que está no R em um formato de arquivo o qual comporte esses meus dados para que outrora eu possa reutilizá-los. Essa função contém alguns parâmetros que são importantes e iremos alterar seus argumentos para termos um banco mais coerente e mais enxuto.

Utilizando o objeto "banco" feito acima, vamos salvá-lo em um arquivo de texto do tipo TXT, facilmente lido em qualquer Bloco de Notas e transferível para qualquer plataforma analítica, da seguinte maneira:

```
write.table(banco,
            file = 'C:/Users/lucianogama/Google Drive/banco.txt',
            sep = '\t',
            row.names = FALSE,
            quote = FALSE,
            na = 'NA')
```

Vamos esmiuçar esse comando? Primeiramente entramos com o nome do objeto "banco" que é o qual contém os nossos dados. O argumento "file = " estabelecerá o destino e o nome do arquivo que será salvo. Observe que nesse exemplo eu estou salvando meu banco na pasta do meu Google Drive instalado no meu computador e o nome que eu dei ao meu arquivo é "banco.txt". Caso você não estabeleça esse caminho da pasta, por padrão o arquivo será salvo na pasta Documentos do Windows; então ficaria apenas "file = banco.txt". Já o argumento "sep = " indica que o separador entre as variáveis no formato texto será o Tab (tabulação). Por padrão, o separador é o espaço (sep = " ") que você pode utilizar também, mas tenha cuidado com a utilização de espaços nos nomes das variáveis, por exemplo. Em "row.names = FALSE", estamos excluindo o nome das linhas (que por padrão é 1, 2, 3, 4, ...), pois raramente nos interessa isso escrito no banco. Com "quote = FALSE" estamos removendo as aspas que a função põe por padrão em todos os dados categóricos e nomes de variáveis. Se você gosta das variáveis separadas por espaço, mesmo que veementemente eu não aconselhe, pôr "quote = TRUE" pode ser fundamental para você. Quanto ao "na = 'NA'", estamos atribuindo os dados faltantes como NA, padrão também do sistema. Há ocasiões em que outras plataformas de bioinformática atribuem dado faltante como "-9", "999", "00", então, se for o caso, isso precisa ser atribuído nesse momento.

2.6 LEITURA DE DADOS (IMPORTAÇÃO)

Já que aprendemos a criar banco de dados estruturados, agora vamos aprender como importá-los para o RStudio. Nem todas as extensões de arquivos o RStudio "abre" de forma igual. Mesmo que pequenas, existem diferenças, porém vamos começar essa conversa com três tipos simples de extensões as quais você pode salvar o seu banco de dados, assim como você pode baixar por aí bancos nesses formatos: *.txt, *.csv e *.tsv.

A extensão TXT é o tipo de arquivo texto mais simples que você possa imaginar para Windows, ou até para outros sistemas operacionais. Nele não há muitas formatações, como negrito e itálico. É o que poderíamos chamar de "texto seco". As extensões CSV e TSV seguem a mesma lógica, porém, no ambiente da bioinformática, é preciso designar o que separam as variáveis, para que o programa importador reconheça cada uma delas e separe-as adequadamente. A CSV separa as variáveis por vírgula, daí seu nome: *comma-separated values* (CSV). A TSV separa as variáveis por tabulação (tecla Tab): *tab-separated values* (TSV).

Para importar nossos bancos de forma mais simples, vá na janela do Environment, clique no botão "import Dataset". Lá você verá algumas opções para importar dados, inclusive de arquivos de Excel, do SPSS, SAS e Stata (plataformas que trabalham com estatística). Veja na Figura 2.5 as possibilidades de imputação de dados de forma interativa. Para os nossos bancos TXT, CSV e TSV, iremos em "From Text (base)" ou "From Text (readr)".

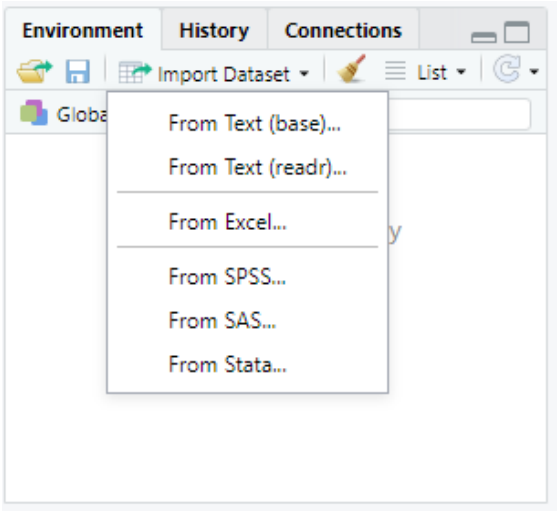


Figura 2.5. Importação de dados.

Indo no menu "From Text (readr)", em "Browse..." você irá buscar o arquivo que deseja e assim que você o escolher, você verá uma prévia em "Data Preview:". Podemos mudar o nome do

banco em “Name”; esse será o nome do objeto. “First Row as Names” marcaremos quando as variáveis estiverem seus nomes na primeira linha do arquivo, o que acontece na maioria das vezes. “Delimiter:” estabelece qual o delimitador entre as variáveis: se vírgula (comma), ponto e vírgula (semicolon), tabulação (Tab), espaço (Whitespace) ou outro (Other...). “Quotes:” atribuímos se colocamos aspas entre as variáveis, o que não é muito comum em termos gerais, mas se houver variáveis com nomes compostos, separados por espaço, talvez seja interessante modificar esse parâmetro. “NA:” estabelecerá o padrão para os dados faltantes, que normalmente é NA, porém, veja acima que há outros padrões na bioinformática.

À medida que você vai estabelecendo os parâmetros, observe que a janela “Code Preview” vai alterando o código (Figura 2.6). O que está ocorrendo na realidade é que o RStudio está escrevendo um comando com base nesses parâmetros que você está modificando. Veja que ele inicia o código carregando o pacote readr, usando a função library(). O código seguinte é para criar o objeto cujo nome você estabeleceu (ou, por padrão, o nome do arquivo selecionado). O terceiro comando é para abrir e visualizar esse objeto no RStudio usando a função View().

Três coisas podem acontecer a partir daí: (1) você clica em “Import” e o seu banco está criado e será aberto para visualização; (2) você copia os comandos da janela “Code Preview” e cola no seu *script* para rodar esse comando depois e clica em “Cancel”; (3) ou, você faz os dois procedimentos: começando por copiar os comandos e colando no seu *script* e depois clicando em “Import”, e assim seu banco está criado e aberto. A vantagem de copiar o código é que você pode modifica-lo depois, caso o banco de dados não tenha sido importado da forma desejada.

Por que não abrir sempre o meu banco pelo “Import Data”? Somente pelo fato de você perder agilidade no dia a dia das suas análises estatísticas. Quando você já tem o código de importação dos dados em um *script*, tudo fica mais rápido. Entretanto, se você gosta mais da parte visual de abrir seus bancos, siga sempre “Import Data” > “From Text (readr)” > modifique parâmetros > “Import” (Figura 2.6).

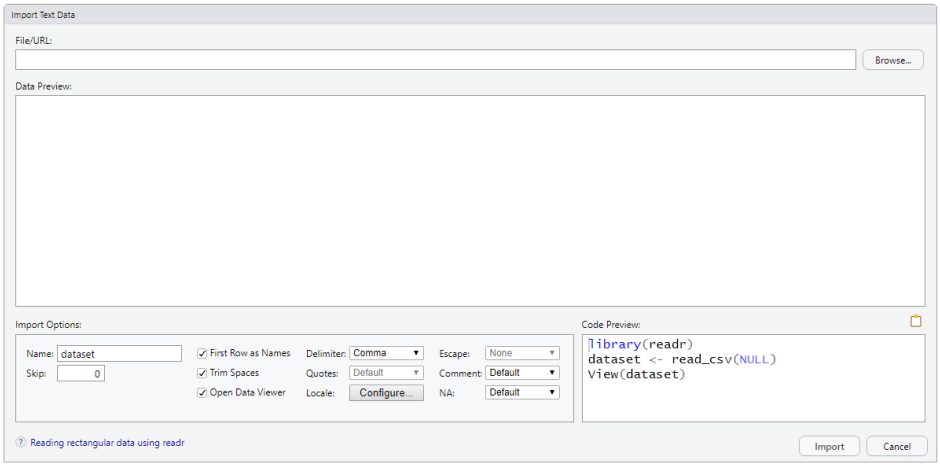


Figura 2.6. Janela do "From Text (readr)".

Com o passar do tempo, estudando e trabalhando no R, pouquíssimas vezes importo os meus bancos de dados pelo "Import". Na maioria das vezes e escrevo o código de importação dos dados no *script* que eu esteja trabalhando. Comumente utilizo a função read.csv(), que serve para a maioria dos tipos de arquivos que eu trabalho. Essa função tem a seguinte sintaxe:

```
read.csv(file, header = TRUE, sep = ",", quote = "\"", dec = ".", fill = TRUE, ...)
```

Nela há alguns parâmetros importantes que são mais usados em arquivos de extensão TXT e CSV. O "file" define o caminho e o arquivo que serão acessados. O "header" especificará se nesse banco de dados tem ou não os nomes das variáveis na primeira linha do documento. O argumento "sep" define qual o caractere que separa as variáveis e seus dados (normalmente espaço, tabulação, vírgula ou ponto e vírgula). O argumento "dec" define qual o caractere que foi usado para dividir a casa decimal dos números contínuos (outra possibilidade de trabalhar com a dualidade vírgula ou ponto na separação de casas decimais).

Dando um exemplo, vamos supor que eu tenha um arquivo de banco de dados salvo em TXT chamado dadosDaPesquisa.txt. Ainda, vamos supor que a primeira linha seja reservada para os nomes das variáveis, como por exemplo: nome do paciente, idade, estado civil, quantidade de filhos, renda familiar. Esses dados estão separados por vírgula (padrão dos arquivos CSV - *Comma Separated Values*). Os decimais dos números foram separados por ponto. E, esse arquivo está na minha pasta Documentos. Vamos gravar esse banco de dados como o objeto chamado "dadosPesquisa" na memória do R. Veja como pode ser escrito o código:

```
dadosPesquisa <- read.csv(
  file = "C:/Users/lucianogama/Documentos/dadosDaPesquisa.txt",
                           sep = ",",
```

```
                          dec = ".",
                          header = TRUE)
```

Não precisaríamos especificar “heard = TRUE” por já ser padrão da função e, nessa situação, poderia ser omitido.

Um bom pacote a ser explorado para importar e exportar banco de dados é o rio (acrônimo de *R input and output*). Busque a documentação do pacote e veja exemplos dos diversos tipos de arquivos que esse pacote permite importar e exportar. Para instalar o rio rode: install.packages('rio').

Essa dica é realmente importante. Para abrir arquivo do Microsoft Excel ou do IBM SPSS, que são bem comuns na área biológica, o melhor meio é usando o pacote rio, com a função import(). Após instalar o pacote, abra a biblioteca. Basicamente, para abrir o arquivo desejado, basta indicar o caminho da pasta com o nome do arquivo.

```
install.packages('rio')
library(rio)
dadosPesquisa <- import('caminho_do_arquivo/pesquisa.xlsx')
```

No caso dos arquivos em Excel talvez seja interessantes o uso de alguns parâmetros. O “range” delimita as células que você deseja abrir do banco. Nem sempre é necessário abrir todo o corpo do banco, então, “range” delimita a extensão das células do Excel que serão abertas. Além disso, o Excel pode conter várias planilhas em um só arquivo. A função import() abrirá por padrão a primeira delas. Para escolher a planilha correta que se deseja, podemos atribuir o nome dela no parâmetro “sheet”.

```
dadosParciaisPesquisa <- import('caminho_do_arquivo/pesquisa.xlsx',
                         range = 'A1:F51',
                         sheet = 'Planilha1')
```

Manipulação de Dados no R

Saiba que uma parte generosa do seu tempo de trabalho analítico será fazendo manipulações dos dados. Por isso esse capítulo é especial. Aqui, entenderemos que você já tem tabulados todos os dados. O que aprenderemos é selecionar variáveis, filtrar classes, criar variáveis novas, juntar diferentes bandos de dados em um só, modificar a codificação de variáveis, enfim. A manipulação de dados nos permite deixar o banco da forma correta para ser analisado ou mesmo para gerar tabelas e gráficos adequadamente.

3.1 O SUPERPACOTE TIDYVERSE

Em capítulos anteriores, conhecemos o que vem a ser pacotes. Contudo, existem pacotes que abrandem muitos outros pacotes "dentro" deles. A gente chama de dependências. Isso é muito útil para maximizar a nossa produtividade, além de deixar a nossa vida mais fácil. Quando temos instalado um desses, que chamarei de superpacotes, teremos a disposição um grande conjunto de ferramentas.

O tidyverse é um desses superpacotes. Quando instalamos ele, na realidade estaremos instalando outros vários pacotes, que são: ggplot2, dplyr, tidyr, readr, purrr, tibble, stringr e o forcats. Com eles, é possível manipular dados, gerar gráficos, modificar variáveis que são do tipo caractere (ou strings), abrir bancos de dados, automatizar comandos, entre muitos outros tipos de funcionalidades.

Para instalar o tidyverse, basta rodar o código abaixo e aguardar a instalação completa.

```
install.packages('tidyverse')
```

Agora, temos que carregar o tidyverse no R para usarmos suas funcionalidades.

```
library(tidyverse)
```

Você verá algo do tipo:

```
-- Attaching packages ------------------------------------------------------ ti-
dyverse 1.3.0 --
v ggplot2 3.3.3     v purrr   0.3.4
v tibble  3.0.6     v dplyr   1.0.4
v tidyr   1.1.2     v stringr 1.4.0
v readr   1.4.0     v forcats 0.5.1
-- Conflicts --------------------------------------------------------- ti-
dyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

Primeiro, é demonstrado a lista de pacotes anexos e suas respectivas versões, além da versão do tidyverse em si (1.3.0, nesse caso). E abaixo, estão listadas as funções em conflito.

Dois ou mais pacotes podem apresentar funções com nomes exatamente iguais e isso gera conflito, uma vez que o R não vai saber exatamente qual é a função de qual pacote você realmente quer usar. Para você definir ao R qual entre as duas ou mais funções você está escolhendo, é necessário iniciar o comando com o nome do pacote que a contém, dois-pontos duas vezes e a função que você deseja. Observe acima que tanto o pacote dplyr quanto o stats contêm a função filter(). Bom, apesar de terem o mesmo nome, elas não são iguais. Geralmente o R dá prioridade ao pacote que foi aberto, mas vamos supor que desejo usar a função filter() do pacote nativo stats. Devemos escrever da seguinte forma:

```
stats::filter()
```

Agora o R sabe que qualquer parâmetro que eu atribuir dentro desse filter() é especificamente o filter() do pacote stats.

Antes de seguir a nossa jornada, vamos abrir um dos nossos bancos de dados que nos servirá de exemplos para a manipulação dos dados. Chamaremos esse banco de "banco1", mas o nome do a ser aberto é o "pesquisa.txt". Esse e outros arquivos estão na pasta do Github chamada BioEpiR, que você pode acessar e baixar através do link: https://github.com/lucianogama/BioEpiR/.

Esse arquivo se trata de um banco fictício de 50 pacientes com seus respectivos dados como ID (identificação), nome (as primeiras letras do nome e sobrenome), ter ou não a doença de estudo, estado brasileiro onde reside, sexo, idade, glicemia (dosagem de glicose em jejum em mg/dL), um índice que chamaremos de X, um índice que chamaremos de Y, escolaridade (variável codificada: 1 = apenas fundamental completo, 2 = ensino médio completo, 3 = ensino superior completo), se tem filhos e quantidade de filhos.

```
#importando arquivo pesquisa.txt
banco1 <- read.csv('C:/_caminho do arquivo em seu computador_/pesquisa.txt',
                   sep = '\t',
                   na = 'NA')
View(banco1)
```

Após rodar o comando acima, veja que em *Environment* teremos criado esse objeto que tem 50 observações e 12 variáveis.

3.2 VINCULANDO UMA PASTA DE TRABALHO

Se você precisa abrir e salvar muitos arquivos, pode ser exaustivo informar ao R toda vez o caminho da pasta onde os arquivos estão ou devem ser salvos. Vai aqui uma dica: use a função setwd() para que o R vincule uma determinada pasta de trabalho. Assim, você não precisará escrever nem se preocupar se está escrevendo o caminho corretamente a todo o momento.

No meu computador pessoal, tenho uma pasta chamada "dados", onde armazeno os bancos de dados que eu utilizo em minhas pesquisas, inclusive o arquivo usado acima, "pesquisa.txt". Rodando o comando abaixo, eu estarei informando ao R a sempre acessar essa pasta. Veja o comando:

```
setwd('C:/Users/lucianogama/dados')
```

Nada de mirabolante ocorrerá no console, porém, veja como podemos imputar os mesmos dados de "pesquisa.txt":

```
#importando arquivo pesquisa.txt de forma mais fácil
banco1 <- read.csv('pesquisa.txt',
                   sep = '\t',
                   na = 'NA')

View(banco1)
```

Perceba que todo o caminho foi suprimido, pois o R já sabe qual a pasta você está trabalhando. Caso haja alguma dúvida em qual pasta o R está "trabalhando", basta rodar o comando de outra função, a getwd():

```
getwd()
```

Você verá no console todo o caminho de pasta:

```
[1] "C:/Users/lucianogama/dados"
```

3.3 OBSERVANDO O BANCO DE DADOS NO R

Nessa primeira parte da manipulação dos dados nós iremos mexer tanto com os pacotes básicos do R quanto também o dplyr. A depender do que iremos manipular, mostrarei mais de uma forma de se fazer. Como dica, baixe e abra o *cheatsheets* do dplyr, pois será um dos mais úteis nesse processo de aprendizado.

Com o nosso banco1 importado, primeiramente, o que devemos fazer é observar os nossos dados. Eles foram importados da forma correta? Preciso me preocupar com a vírgula, se lembra?

Será que o delimitador que separam as variáveis estava correto e, portanto, o R separou as variáveis corretamente? É sempre bom dar uma olhada geral no banco logo após a importação.

Como visto acima, podemos ver isso utilizando a função View(). Uma aba com os dados tabulados será aberta. Mas, podemos utilizar outras duas funções que dá para observar os dados de forma genérica, simples e resumida: head() e tail(). A head() mostrará a parte inicial dos dados (cabeça), enquanto tail() mostrará a parte final dos dados no console (cauda).

```
head(banco1)
   ID  nome doenca estado      sexo idade glicemia indice_x indice_y
1 511  SGSG    sim     ES masculino    46       97     0.99     0.84
2 512 WQBTS    sim     RS  feminino    38       84     0.76     0.32
3 513    FR    nao     AM masculino    20       99     0.73     0.84
4 514   BGY    nao     RR  feminino    47       81     0.55     0.49
5 515  NOPS    sim     RO  feminino    40       87     0.91     0.88
6 516  WEGB    nao     BA masculino    70       71     0.27     0.29
```

```
tail(banco1)
    ID  nome doenca estado      sexo idade glicemia indice_x indice_y
45 555 SXASA    sim     AC masculino    49      109     0.95     0.90
46 556   AFB    sim     MT masculino    82       99     0.19     0.62
47 557  ZOPR    nao     PR  feminino    50      113     0.92     0.93
48 558  FTHS    nao     RJ masculino    29      106     0.16     0.09
49 559  GGWA    sim     SE  feminino    37       94     0.85     0.58
50 560  AFMG    sim     MS masculino    88       91     0.63     0.59
```

Perceba apenas que eu suprimi as três últimas variáveis para caber aqui nas páginas do livro, porém, no console serão mostradas as seis primeiras observações e todas as variáveis do objeto, por padrão.

Porém, podemos mudar isso. Modificando o parâmetro "n", podemos escolher a quantidade de linhas que serão retornadas. Vamos ver somente a três primeiras:

```
head(banco1, n = 3)
   ID  nome doenca estado      sexo idade glicemia indice_x indice_y
1 511  SGSG    sim     ES masculino    46       97     0.99     0.84
2 512 WQBTS    sim     RS  feminino    38       84     0.76     0.32
3 513    FR    nao     AM masculino    20       99     0.73     0.84
```

Agora as três últimas:

```
tail(banco1, n = 3)
    ID nome doenca estado      sexo idade glicemia indice_x indice_y
48 558 FTHS    nao     RJ masculino    29      106     0.16     0.09
49 559 GGWA    sim     SE  feminino    37       94     0.85     0.58
50 560 AFMG    sim     MS masculino    88       91     0.63     0.59
```

Podemos fazer coisa semelhante usando colchetes após o nome do objeto, sendo ele data frame ou matriz. Dentro dos colchetes podemos escolher o intervalo de linhas e o intervalo de colunas que eu desejo que o R exiba no *Console*.

Exibindo as dez primeiras linhas (observações) e as cinco primeiras variáveis, podemos escrever o código:

```
banco1[1:10,1:5]
    ID  nome doenca estado      sexo
1  511  SGSG    sim     ES masculino
2  512 WQBTS    sim     RS  feminino
3  513    FR    nao     AM masculino
4  514   BGY    nao     RR  feminino
5  515  NOPS    sim     RO  feminino
6  516  WEGB    nao     BA masculino
7  517 FRTGS    nao     PA  feminino
8  518 EFGTR    sim     AP  feminino
9  519  SWPO    sim     SE masculino
10 520  LGSG    nao     PE masculino
```

Há a possibilidade de verificar uma variável, ou conjunto de variáveis, sem necessariamente escrever a vírgula. Vamos mostrar as três primeiras variáveis:

```
banco1[1:3]
    ID    nome doenca
1  511    SGSG    sim
2  512   WQBTS    sim
3  513      FR    nao
4  514     BGY    nao
5  515    NOPS    sim
6  516    WEGB    nao
7  517   FRTGS    nao
8  518   EFGTR    sim
9  519    SWPO    sim
10 520    LGSG    nao
```

O mesmo ocorre quando eu coloco a vírgula. Então, para a visualização apenas das colunas, a vírgula é opcional. Perceba:

```
banco1[,1:3]
    ID    nome doenca
1  511    SGSG    sim
2  512   WQBTS    sim
3  513      FR    nao
4  514     BGY    nao
5  515    NOPS    sim
6  516    WEGB    nao
7  517   FRTGS    nao
8  518   EFGTR    sim
9  519    SWPO    sim
10 520    LGSG    nao   (...)
```

3.4 OCORRÊNCIA DE DADOS FALTANTES

Já falamos que bancos de dados podem vim com dados faltantes, porém, não é das coisas mais desejáveis a ocorrer. Precisamos nos preocupar com isso, pois dados faltantes podem atrapalhar nossas análises e perder poder de teste e, quando possível, precisaremos trata-los.

Também já falamos das possibilidades de como se mostram valores faltantes no meio da bioinformática. Então, vou considerar aqui apenas a forma NA, que é a forma padrão de se mostrar dados faltantes no R. Para sabermos se no nosso banco tem algum NA, usaremos a função is.na(). A função table() nos ajudará a tabular o quantitativo de dados faltantes. Veja abaixo:

```
faltantes <- table(is.na(banco1))
faltantes
```

O R nos retornará:

```
FALSE  TRUE

  582    18
```

Ou seja, FALSE indicam os dados que não são NA, enquanto TRUE serão os dados faltantes. Então, obviamente temos 18 células as quais não têm nenhum valor. Até aqui ainda não sabemos em quais variáveis ocorre isso. Se rodarmos View(banco1), poderemos observar todo o banco e ver onde há NA.

Vamos ver a porcentagem de dados faltantes em nosso banco de dados. Para isso, vamos acessar os resultados dentro do objeto "faltantes" que criamos com colchetes duplos. Observe:

```
faltantes[[1]]       #resultado de dados não NA
[1] 582

faltantes[[2]]       #resultado de dados NA
[1] 18
```

Só a nível de curiosidade, perceba que faltantes[1] é diferente de faltantes[[1]] ao que o R retorna de resultado.

Para calcular a porcentagem, precisaremos do total de dados, que é 582 + 18 = 600. Então, podemos calcular a porcentagem de NA de modo geral utilizando a fórmula:

$$P(\text{NA}) = \frac{números\ de\ NAs}{total\ de\ dados}$$

No R:

```
#nrow retorna o número de linhas
#ncol retorna o número de colunas
nrow(banco1) * ncol(banco1)       #para saber o total de células do banco
porcentagemNA <- faltantes[[2]] / 600
porcentagemNA
[1] 0.03
```

O valor retornado foi 0,03, o que corresponde a 3% de dados faltantes.

Para sabermos se uma determinada **variável** apresenta NA, podemos especificá-la logo após cifrão ($) que vem depois do nome do objeto. Então:

```
faltantes_var <- table(is.na(banco1$quant_filhos))
FALSE  TRUE
   32    18
```

Pelo visto, o que temos de dados faltantes está contido na variável que quantifica o número de filhos (quant_filhos). Para calcularmos a porcentagem de perda de dados dentro da variável, devemos fazer o mesmo cálculo acima, mudando o total para o valor total de observações da variável.

```
#length() retornará o "comprimento" da variável, ou seja, o valor 50.
faltantes_var[[2]] / length(banco1$quant_filhos)
[1] 0.36
```

O valor retornado foi 0.36, ou seja, 36% dos dados da variável quant_filhos foram perdidos. Nem sempre dado faltante quer dizer que não temos informação. Criticamente falando, perceba que para essa variável em específico apresentar NA quer dizer que a pessoa não tem filho algum; analise junto com a variável "filhos" que é categórica. Alguns bancos de dados podem chegar assim até você e você precisa compreender cada variável para trabalha-la da melhor forma possível. Nesse caso específico, acredito que ter tabulado NA como não ter filho algum é um equívoco, seria melhor ter posto o valor zero.

É difícil estabelecer uma porcentagem máxima aceitável de dados faltantes para considerar a variável analisável. Na literatura há indícios que um banco de dados pobre de informação é quando ele tem 5% a 25% de valores faltantes (SCHEFFER, 2002). Mas isso depende muito da importância da variável na questão da pesquisa. Para tratar isso, há meios de lidar com os NA.

3.5 TRATANDO DADOS FALTANTES

Como visto, dados faltantes podem ser um problema para uma gama de análises estatísticas. Há diversos métodos para resolvermos esse problema, que é, obviamente, pré-analítico. Uma forma bruta é meramente excluindo as variáveis e indivíduos que apresentem pelo menos um dado

faltante (ou, melhor que seja, mais de 10% de dados faltantes). Entenda que porcentagem é uma coisa relativa. Se tivermos 10 observações e 10% são ausentes, será 1 única observação que falta em nosso banco. Porém, se tivermos 1 milhão de observações, 10% será 100 mil observações que não conhecemos, que perdemos a informação. É muito dado perdido. Dessa forma, cabe ao analista refletir sobre os prós e contras dentre os diversos meios de se tratar dados faltantes em sua realidade analítica.

Lembrando a regra suprema: o banco tratado deve ser o mais fidedigno possível ao banco original, portanto, à realidade. Saber isso é complexo, porque depende do conhecimento e experiência do analista sobre o assunto que ronda os dados.

Aqui, abordaremos métodos simples de tratamento por se tratar de um livro para iniciantes no tema. Mas fica a sugestão de você ir além do escopo do livro se você almeja trabalhar com ciência de dados.

Pois bem, o primeiro meio de tratamento abordado será usando a função na.omit() do R. Ela pode ser muito útil, porém, ela "corta o mal pela raiz". A função eliminará toda linha (portanto, toda observação) que houver pelo menos um dado faltante. Vamos exemplificar esse método criando o simples banco de dados:

```
dados <- data.frame(
  var1 = c(9, 2, NA, 7, 5, NA, 8, 1, NA, 8),
  var2 = c('dor', 'sem', 'dor', 'dor', 'sem', 'dor', 'sem', NA, 'dor', NA)
)

dados
   var1 var2
1     9  dor
2     2  sem
3    NA  dor
4     7  dor
5     5  sem
6    NA  dor
7     8  sem
8     1 <NA>
9    NA  dor
10    8 <NA>
```

Os dados cujo valor for NA indicará ao R que esse dado está perdido. Dessa forma, é muito importante que no momento de coleta de dados o termo "NA" não seja nenhuma opção de caracterização de dados a não ser de perda de informação. Do que estou falando? Vamos supor que estamos coletando dados sobre idosos e que queira tabular a informação de que ele não está sendo acompanhado por um geriatra. Evite pôr NA no intuito de informar "Não Acompanhado", por exemplo.

Voltando ao nosso pequeno banco de dados, veja que há 5 dados que não foram possíveis ficticiamente de serem obtidos. Rodando na.omit() nesse objeto, perceba que perdemos 5 linhas:

```
na.omit(dados)

  var1 var2
1    9  dor
2    2  sem
4    7  dor
5    5  sem
7    8  sem
```

Você pode criar um novo banco usando:

```
dados_semNA1 <- na.omit(dados)
dados_semNA1
  var1 var2
1    9  dor
2    2  sem
4    7  dor
5    5  sem
7    8  sem
```

Perceba que nessa situação perdemos metade do banco original. Talvez, em algumas situações, não é o melhor a se fazer.

Vamos escolher um outro meio, o da imputação de dados. Se não conhecemos o valor faltante, pode ser razoável trocá-lo pela média (conversaremos mais sobre médias mais à frente, mas saiba que é um valor que generaliza os dados de uma variável numérica). Assim, podemos escrever o comando usando toda a bagagem do tidyverse:

```
dados_semNA2 <- dados %>%
  mutate(var1_imp = ifelse(is.na(var1) == TRUE, mean(var1, na.rm = T), var1))
dados_semNA2
       var1 var2
1  9.000000  dor
2  2.000000  sem
3  5.714286  dor
4  7.000000  dor
5  5.000000  sem
6  5.714286  dor
7  8.000000  sem
8  1.000000 <NA>
9  5.714286  dor
10 8.000000 <NA>
```

Vamos destrinchar o que houve aqui:

- "dados_semNA2 <-" estamos criando um novo objeto;
- "dados %>%" eu estou mostrando ao R que eu quero mexer nesse objeto/banco de dados e que ele siga em frente ao próximo comando;

- "mutate()" é uma função que cria novas variáveis (do pacote dplyr, que vem junto com o tidyverse);
- "var1" é o nome da variável que eu quero imputar a média nos dados faltantes;
- "ifelse()" é uma função condicional: se... então...;
- "is.na()" é a função do R que vai verificar se um determinado dado é NA;
- "mean(, na.rm = TRUE)" é uma função que calcula a média de uma variável. Quando usamos o argumento "na.rm = TRUE" estamos informando que queremos excluir os NA no cálculo da média. Se não excluímos, o R retornará erroneamente NA.

Perceba que a variável original foi alterada. Nos valores que continha NA agora contém a média dos valores originais (aproximadamente 5,71). Outra forma de imputar os dados faltantes pela média seria usando o comando seguinte:

```
dados$var1 <- ifelse(is.na(dados$var1) == TRUE, mean(dados$var1,
                     na.rm = TRUE), dados$var1)
```

Seguindo em frente, perceba que a variável "var2" também tem NA, porém ela é categórica. A média não funciona para ela, mas a frequência sim. Um método que pode ser utilizado para contornar valores categóricos faltantes é imputar a classe que mais se repete.

Primeiramente, vamos criar a função chamada "mais_repete" para calculá-la posteriormente. O R não tem essa função previamente criada nos seus pacotes básicos, apenas pode nos mostrar, como é o caso da função table(). Não se preocupe muito com a complexidade da função que criaremos aqui nesse momento, porque nós vamos falar muito sobre ao longo do livro. Basta redigir o comando no seu *script* de treinamento.

```
#criando a função mais_repete
mais_repete <- function(x){
  uniq <- unique(x)
  uniq[which.max(tabulate(match(x, uniq)))]
}
```

Vamos imputar os dados faltantes segundo o valor mais frequente que acharmos da variável var2.

```
#verificando a classe que se repete
table(dados$var2)

dor sem
  5   3

#imputando dados faltantes em var2 (categórica)
dados$var2 <- ifelse(is.na(dados$var2) == TRUE, mais_repete(dados$var2),
                     dados$var2)
```

```
dados
       var1 var2
1  9.000000  dor
2  2.000000  sem
3  5.714286  dor
4  7.000000  dor
5  5.000000  sem
6  5.714286  dor
7  8.000000  sem
8  1.000000  dor
9  5.714286  dor
10 8.000000  dor
```

Pronto, agora temos o banco completo. Algumas análises estatísticas só rodam se o banco estiver completo, outras isso já não necessário. Métodos para imputar dados em dados faltantes existem vários disponíveis. Daria um livro a parte, como na verdade já existe.

Existem diversos pacotes que tratam da imputação de dados, como o mice (*Multivariate Imputation by Chained Equations*). Ele é robusto e especializado nessas situações. Nele, há vários métodos de imputação. Sugiro o estudo da sua documentação se dados faltantes para você for realmente um problema.

3.6 SELECIONANDO LINHAS E COLUNAS

Vamos abrir o banco de dados íris usando o comando `data(iris)` no R. Vamos visualizar rapidamente o banco rodando o comando `View(iris)`. Nele há quatro variáveis numéricas e uma variável categórica (*Species*). Para selecionar as três primeiras colunas em um outro *data frame*, podemos usar o comando:

```
iris1 <- iris[, 1:3]
iris1
    Sepal.Length Sepal.Width Petal.Length
1            5.1         3.5          1.4
2            4.9         3.0          1.4
3            4.7         3.2          1.3
4            4.6         3.1          1.5
5            5.0         3.6          1.4
6            5.4         3.9          1.7
7            4.6         3.4          1.4
8            5.0         3.4          1.5
9            4.4         2.9          1.4
10           4.9         3.1          1.5
11           5.4         3.7          1.5
12           4.8         3.4          1.6
13           4.8         3.0          1.4      (...)
```

Dentro dos colchetes, após a vírgula estamos selecionando a sequência de colunas desejada. Perceba que a seleção é também uma exclusão, ou seja, estamos excluindo as duas últimas colunas.

Vamos agora selecionar as dez primeiras linhas, ou seja, selecionarmos as dez primeiras observações:

```
iris2 <- iris[1:10, ]
iris2
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4         0.2  setosa
2           4.9         3.0          1.4         0.2  setosa
3           4.7         3.2          1.3         0.2  setosa
4           4.6         3.1          1.5         0.2  setosa
5           5.0         3.6          1.4         0.2  setosa
6           5.4         3.9          1.7         0.4  setosa
7           4.6         3.4          1.4         0.3  setosa
8           5.0         3.4          1.5         0.2  setosa
9           4.4         2.9          1.4         0.2  setosa
10          4.9         3.1          1.5         0.1  setosa
```

Veja que foi mantida todas as colunas, porém, temos agora somente os dez primeiros indivíduos.

Vamos agora selecionar as cinco primeiras observações e as duas primeiras colunas:

```
iris3 <- iris[1:5, 1:2]
iris3
  Sepal.Length Sepal.Width
1          5.1         3.5
2          4.9         3.0
3          4.7         3.2
4          4.6         3.1
5          5.0         3.6
```

Se o meu desejo é selecionar apenas algumas linhas e colunas, sem que estejam ordenadas, precisamos usar o comando c(), separando por vírgula o que se deseja. Vamos selecionar a linha 1 e a linha 64, juntamente com a coluna 2 e a coluna 5. Veja:

```
iris4 <- iris[c(1,64) , c(2,5)]
iris4
   Sepal.Width    Species
1          3.5     setosa
64         2.9 versicolor
```

Existem outras formas de selecionar vaiáveis em um banco secundário. Quando eu conheço perfeitamente os nomes das variáveis, prefiro utilizar a função select() do pacote dplyr, que está embutido no tidyverse. Como parâmetro da função, basta listar dentro dela todas as variáveis que deseja selecionar.

Mais acima vimos o uso de um pipe (%>%) que é específico do pacote magrittr (que também faz parte do tidyverse), mas é utilizado por diversas outras funções. Esse pequeno código diz ao R que o que estiver antes dele é para ser usado no código seguinte. Isso acaba otimizando nossa interação com as funções, deixando mais limpo o código. Fora isso, evita com que a gente crie a

todo momento objetos que ocupem demais a memória RAM do computador. Vamos utilizá-lo para selecionar as variáveis “Sepal.Length” e “Sepal.Width” do banco iris, veja como:

```
 iris5 <- iris %>%
  select(Sepal.Length,Sepal.Width)

iris5
    Sepal.Length Sepal.Width
1            5.1         3.5
2            4.9         3.0
3            4.7         3.2
4            4.6         3.1
5            5.0         3.6
6            5.4         3.9
7            4.6         3.4   (...)
```

3.7 FILTRANDO O BANCO DE DADOS

Filtrar não deixa de ser um método de seleção, contudo, estamos selecionando observações. Primeiramente, vamos ver como podemos filtrar dados comandos básicos do R e depois usando funções do pacote dplyr, que está incluso no tidyverse.

Para exemplificar, no banco de dados iris vamos filtrar os indivíduos que apresentam sépalas maiores do que 7 centímetros usando o comando:

```
iris_filtro <- iris[iris$Sepal.Length > 7,]

iris_filtro
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
103          7.1         3.0          5.9         2.1 virginica
106          7.6         3.0          6.6         2.1 virginica
108          7.3         2.9          6.3         1.8 virginica
110          7.2         3.6          6.1         2.5 virginica
118          7.7         3.8          6.7         2.2 virginica
119          7.7         2.6          6.9         2.3 virginica
123          7.7         2.8          6.7         2.0 virginica
126          7.2         3.2          6.0         1.8 virginica
130          7.2         3.0          5.8         1.6 virginica
131          7.4         2.8          6.1         1.9 virginica
132          7.9         3.8          6.4         2.0 virginica
136          7.7         3.0          6.1         2.3 virginica
```

Você terá o mesmo resultado usando o comando `filter()` do dplyr rodando o comando abaixo. Só não esqueça de abrir a biblioteca tidyverse antes: `library(tidyverse)`.

```
iris_filtro <- filter(iris, Sepal.Length > 7)

iris_filtro
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species

103          7.1         3.0          5.9         2.1 virginica
106          7.6         3.0          6.6         2.1 virginica
```

```
108          7.3         2.9          6.3         1.8 virginica
110          7.2         3.6          6.1         2.5 virginica
118          7.7         3.8          6.7         2.2 virginica
119          7.7         2.6          6.9         2.3 virginica
123          7.7         2.8          6.7         2.0 virginica
126          7.2         3.2          6.0         1.8 virginica
130          7.2         3.0          5.8         1.6 virginica
131          7.4         2.8          6.1         1.9 virginica
132          7.9         3.8          6.4         2.0 virginica
136          7.7         3.0          6.1         2.3 virginica
```

Existem nesse banco três tipos de espécies de flor: setosa, versicolor e virginica. Vamos filtrar apenas as que são setosa. Devemos ficar atentos à vírgula.

```
iris_filtro <- iris[iris$Species == "setosa",]

iris_filtro
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4         0.2  setosa
2           4.9         3.0          1.4         0.2  setosa
3           4.7         3.2          1.3         0.2  setosa
4           4.6         3.1          1.5         0.2  setosa
5           5.0         3.6          1.4         0.2  setosa
6           5.4         3.9          1.7         0.4  setosa
7           4.6         3.4          1.4         0.3  setosa
8           5.0         3.4          1.5         0.2  setosa  (...)
```

Você terá o mesmo resultado usando o comando filter() do dplyr com esse comando:

```
iris_filtro <- filter(iris, Species == 'setosa')

iris_filtro
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4         0.2  setosa
2           4.9         3.0          1.4         0.2  setosa
3           4.7         3.2          1.3         0.2  setosa
4           4.6         3.1          1.5         0.2  setosa
5           5.0         3.6          1.4         0.2  setosa
6           5.4         3.9          1.7         0.4  setosa
7           4.6         3.4          1.4         0.3  setosa  (...)
```

Mesmo resultado teremos usando o pipe (%>%):

```
iris_filtro <- iris %>%
  filter(Species == 'setosa')

iris_filtro
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4         0.2  setosa
2           4.9         3.0          1.4         0.2  setosa
3           4.7         3.2          1.3         0.2  setosa
4           4.6         3.1          1.5         0.2  setosa
5           5.0         3.6          1.4         0.2  setosa
6           5.4         3.9          1.7         0.4  setosa
7           4.6         3.4          1.4         0.3  setosa  (...)
```

3.8 CRIANDO E MODIFICANDO VARIÁVEIS

Vamos criar um pequeno banco de dados, inicialmente com duas variáveis: altura de cinco pessoas (em metros) e seus respectivos pesos (em quilogramas). Vamos ao código.

```
df = data.frame(
  altura = c(1.73, 1.59, 1.92, 1.83, 1.68),
  peso = c(75, 57, 124, 91, 64)
)
```

Eu espero que nesse momento você perceba que há formas diferentes de se fazer a mesma coisa (no momento estou falando de criação de objetos usando apenas o sinal de igualdade). Vamos calcular o IMC (Índice de Massa Corpórea) desses cinco indivíduos e atribuir esses valores a uma variável que até então não existia no banco. O IMC se dá com a divisão do peso pelo quadrado da altura. Vamos ao código disso:

```
df$imc <- df$peso / (df$altura)^2

df
  altura peso      imc
1   1.73   83 27.73230
2   1.59   57 22.54658
3   1.92  124 33.63715
4   1.83   91 27.17310
5   1.68   69 24.44728
```

Dessa maneira, acabamos de criar uma nova variável que ficará registrada no objeto *data frame* que criamos. Se você não definir df$imc <- antes do cálculo, aparecerá os resultados, mas não serão armazenados no objeto como uma variável.

```
df$peso/(df$altura)^2
[1] 27.73230   22.54658   33.63715   27.17310   24.44728
```

Além disso, você já deve ter percebido que para criar ou chamar uma variável de um *data frame* ou matriz, no código basta escrever o nome do banco, pôr o caractere cifrão e depois o nome da variável.

O valor do IMC é dado em kg/m² e o seu valor é de grande valia para os nutricionistas e médicos endocrinologistas, pois assim eles podem avaliar se a pessoa está com o peso normal ou não. A grosso modo, pessoas com IMC acima de 25 estão em sobrepeso. Então, vamos usar esse limite para classificar nossa amostragem. Para isso, vamos criar uma nova variável categórica usando a função ifelse() no R. Essa função simplifica nossa vida, pois if (traduzindo do inglês "se") e else ("senão") juntas resumem dois comandos condicionais que poderiam ser usados separadamente. O código que criaremos será lido pelo R mais ou menos da seguinte forma: crie uma

variável chamada classe no bando de dados chamado “df”, sendo que, se a variável imc foi menor do que 25, preencha a nova variável com “abaixo”, senão preencha com “acima”. Veja:

```
df$classe <- ifelse(df$imc < 25, 'abaixo', 'acima')

df
  altura peso      imc classe
1   1.73   83 27.73230  acima
2   1.59   57 22.54658 abaixo
3   1.92  124 33.63715  acima
4   1.83   91 27.17310  acima
5   1.68   69 24.44728 abaixo
```

Acabamos de fazer uma dicotomia: abaixo e acima. Para a área biológica em geral, as dicotomias são muito convenientes e constantemente usamos elas para tentarmos entender melhor certos desfechos (apesar de que a vida possa ser mais complexa do que uma simples dicotomia). De qualquer forma, é um ótimo começo analítico.

Há no âmbito da estatística o uso recorrente da dicotomia 0 e 1. Por exemplo, estar saudável é 0 e estar doente é 1. Isso nos ajuda em várias situações e não se assuste se alguns programas de estatística e algumas análises necessitarem a rigor desse tipo de construção. Portanto, vamos exercitar isso usando novamente o ifelse().

Aportuguesando o que vamos escrever em R é: crie uma nova variável chamada di em df, sendo que, se a variável classe for abaixo, coloque 0 na nova variável, senão, coloque 1.

```
df$di <- ifelse(df$classe == 'abaixo', 0, 1)

df
  altura peso      imc classe di
1   1.73   83 27.73230  acima  1
2   1.59   57 22.54658 abaixo  0
3   1.92  124 33.63715  acima  1
4   1.83   91 27.17310  acima  1
5   1.68   69 24.44728 abaixo  0
```

Essa transformação de variáveis estabelecendo 0 ou 1 é chamada de *dummy*. Análises de predição mais complexas usam muito essa conformação de variável porque é um dos métodos de transformar variáveis categóricas em variáveis numéricas. Entenda que há análises estatísticas que só podem ser feitas com variáveis estritamente numéricas. A “dummyzação”, se é que podemos fazer esse neologismo, é uma das etapas mais comuns para esse fim. Não aprofundaremos essa transformação, mas saiba que há pacotes específicos disponíveis para transformar variáveis em *dummy* automaticamente com códigos simples, muito úteis para modificar bancos de dados extensos. Já imaginou ter que transformar cem mil variáveis qualitativas em variáveis *dummy*? Seriam

cem mil linhas de comandos como a de cima. E se fosse um milhão de variáveis? Totalmente inviável escrever um código para cada variável e ser uma pessoa feliz nessa vida.

Se você for necessitar desse tipo de transformação em bancos muito grandes, sugiro ir ao site oficial do R e procurar por pacotes que te satisfaça. São exemplos: fastDummies, dummy e makedummies. Mas, se o trabalho for a transformação de uma, duas ou três variáveis, você já sabe como fazer e vamos seguir adiante.

3.9 COMBINANDO DADOS DE BANCOS DIFERENTES (*MERGE*)

Construir um banco de dados nem sempre é simples. Por vezes precisamos juntar bancos de dados diferentes. Vamos dar aqui um exemplo muitíssimo simples: supondo que solicitamos que dois alunos de fisioterapia tabulassem dados dos seus pacientes atendidos num determinado hospital universitário. Cada um iria criar um banco próprio, porém, com as mesmas variáveis: iniciais do paciente (id), data do atendimento, tipo de reabilitação e idade.

```
aluno1 <- data.frame(
  id = c('FSTE', 'HESG', 'MCTEAF', 'LGSG', 'MAJM'),
  data = c('04/02/2022', '09/02/2022', '19/05/2022', '14/08/2022', '30/09/2022'),
  reabili = c('fratura madibular', 'torção joelho', 'amputação', 'asma',  'distúrbio
cardíaco'),
  idade = c(26, 15, 74, 42, 68)
)
```

```
aluno2 <- data.frame(
  id = c('ATFG', 'THB', 'NEDGHD', 'PHSW', 'KJSF'),
  data = c('07/02/2022', '08/02/2022', '16/04/2022', '21/05/2022', '27/06/2022'),
  reabili = c('DPOC', 'escoliose', 'pós-cirúrgico mão D', 'COVID-19',  'neuro'),
  idade = c(54, 26, 30, 41, 65)
)
```

Supondo que o professor-orientador desses dois alunos solicitou a junção desses dois bancos, como poderíamos proceder facilmente no R? O primeiro a se fazer é verificarmos se estão padronizados. As variáveis apresentam os mesmos nomes? Escritas exatamente da mesma forma? Estão na mesma sequência? Se sim, nesses condições podemos usar a função rbind() da seguinte forma:

```
todos <- rbind(aluno1, aluno2)

todos
```

```
       id       data             reabili idade
1    FSTE 04/02/2022   fratura madibular    26
2    HESG 09/02/2022       torção joelho    15
3  MCTEAF 19/05/2022           amputação    74
4    LGSG 14/08/2022               asma    42
5    MAJM 30/09/2022  distúrbio cardíaco    68
6    ATFG 07/02/2022               DPOC    54
7     THB 08/02/2022           escoliose    26
8  NEDGHD 16/04/2022 pós-cirúrgico mão D    30
9    PHSW 21/05/2022            COVID-19    41
10   KJSF 27/06/2022               neuro    65
```

Bind em inglês significa ligar, vincular. O prefixo "r" da função vem de *rows* (traduzindo: linhas). Como você já deve ter captado, essa função junta dois objetos (vetores, matrizes ou *data frames*) através das linhas, um acima do outro.

Já, com a função cbind() podemos vincular dois bancos de dados (vetores, matrizes ou *data frames*) com base em suas colunas, um do lado do outro. Como exemplo, vamos supor que o orientador necessite adicionar mais duas variáveis que foram tabuladas num momento depois: o número de sessões feitas e se o paciente finalizou o tratamento ou não.

```
add <- data.frame(
  sessoes = c(5, 7, 4, 4, 8, 7, 4, 6, 5, 9),
  termino = c('sim', 'sim', 'nao', 'sim', 'sim', 'sim', 'nao', 'sim', 'nao', 'sim')
)
```

Usando cbind() podemos juntar essas duas variáveis ao banco total anterior:

```
todosFinal <- cbind(todos, add)

todosFinal
       id       data             reabili idade sessoes termino
1    FSTE 04/02/2022   fratura madibular    26       5     sim
2    HESG 09/02/2022       torção joelho    15       7     sim
3  MCTEAF 19/05/2022           amputação    74       4     nao
4    LGSG 14/08/2022               asma    42       4     sim
5    MAJM 30/09/2022  distúrbio cardíaco    68       8     sim
6    ATFG 07/02/2022               DPOC    54       7     sim
7     THB 08/02/2022           escoliose    26       4     nao
8  NEDGHD 16/04/2022 pós-cirúrgico mão D    30       6     sim
9    PHSW 21/05/2022            COVID-19    41       5     nao
10   KJSF 27/06/2022               neuro    65       9     sim
```

É muito importante que você perceba que a sequência dos valores precisa coincidir com a sequência dos respectivos pacientes. Dessa forma, esses dois métodos muito simples precisam ser utilizados com uma certa cautela, pois geralmente bancos de dados robustos não vêm tão bem estruturados assim.

Os arquivos aluno1, aluno2.txt e add.txt estão no Github (https://github.com/luciano-gama/BioEpiR) caso você deseje baixar esses bancos de dados separadamente.

Existe uma outra forma, e mais coerente aos bancos mais complexos, de juntar e combinar valores de dois bancos de dados diferentes usando funções do pacote dplyr. São quatro funções: left_join(), right_join(), inner_join() e full_join(). Todas elas trabalham com minimamente os seguintes parâmetros: "x", "y" e "by". O parâmetro "x" é o banco da esquerda, digamos assim, o "y" é o banco da direita e o "by" define a variável a qual os dois bancos têm em comum para que o R combine adequadamente eles dois. Por isso é importante sempre utilizar uma variável identificadora de cada indivíduo. Normalmente a chama de ID ou algo do tipo. Em resumo, no geral temos a sintaxe:

```
_join(x=primeiro_banco, y=segundo_banco, by='variável_de_identificação')
```

Mas qual a diferença entre essas quatro?

- left_join() – matem todos os valores do objeto x e junta com os valores de y. As observações que não estiverem presentes em y serão tratadas como NA.
- right_join() – matem todos os valores de y e junta com os valores de x. As observações que não estiverem presentes em x serão tratadas como NA.
- inner_join() – serão mantidos todos os valores de x e y que combinam conjuntamente. Ou seja, as observações que são exclusivas de x, ou de y, serão descartadas. Essa função é uma das mais utilizadas, pois reteremos os indivíduos que apresentam dados no banco x e y.
- full_join() – serão mantidos definitivamente todos os valores ,e os valores que não combinarem serão definidos como NA.

Vamos praticar? Primeiro, vamos retomar o objeto "todosFinal" que criamos acima. Certifique-se que ele está criado ou baixo-o no Github e importe para o R. Depois, crie (ou baixe) o *data frame* abaixo, que será o banco de dados onde acrescentaremos as variáveis adicionais de cada indivíduo.

```
imcDor <- data.frame(
  id = c('FSTE', 'HESG', 'LGSG', 'MAJM', 'SVTO', 'NEDGHD'),
  imc = c(24.82, 25.21, 30.42, 21.29, 25.01, 40.11),
  dor = c(8, 9, 2, 5, 2, 7)
)
```

Perceba que a variável "id", que identifica os pacientes, está presente nesse novo banco também. Ela é a chave para relacionarmos os dados entre os dois bancos. O que há de novo são os dados de IMC (Índice de Massa Corpórea) e de uma escala de dor (que vai de 0 a 10 – EVA [Escala Visual de Analógica]) que não haviam sido tabulados no primeiro momento. Quero chamar

atenção também ao penúltimo indivíduo desse novo banco (SVTO), pois ele é um paciente "novo", o qual não havia sido listado no banco "todosFinal".

Pois bem, vamos testar todas as quatro funções e, por fim, perceber onde elas são diferentes entre si.

```
merge1 <- left_join(todosFinal, imcDor, by = "id")
merge1
       id       data              reabili idade sessoes termino   imc dor
1    FSTE 04/02/2022   fratura madibular    26       5     sim 24.82   8
2    HESG 09/02/2022       torção joelho    15       7     sim 25.21   9
3  MCTEAF 19/05/2022           amputação    74       4     nao    NA  NA
4    LGSG 14/08/2022                asma    42       4     sim 30.42   2
5    MAJM 30/09/2022  distúrbio cardíaco    68       8     sim 21.29   5
6    ATFG 07/02/2022                DPOC    54       7     sim    NA  NA
7     THB 08/02/2022           escoliose    26       4     nao    NA  NA
8  NEDGHD 16/04/2022 pós-cirúrgico mão D    30       6     sim 40.11   7
9    PHSW 21/05/2022            COVID-19    41       5     nao    NA  NA
10   KJSF 27/06/2022               neuro    65       9     sim    NA  NA
```

Veja que todos os indivíduos de x foram mantidos, porém, os valores que não há correspondência com y foram tratados como NA.

```
merge2 <- right_join(todosFinal, imcDor, by = "id")
merge2
      id       data              reabili idade sessoes termino   imc dor
1   FSTE 04/02/2022   fratura madibular    26       5     sim 24.82   8
2   HESG 09/02/2022       torção joelho    15       7     sim 25.21   9
3   LGSG 14/08/2022                asma    42       4     sim 30.42   2
4   MAJM 30/09/2022  distúrbio cardíaco    68       8     sim 21.29   5
5 NEDGHD 16/04/2022 pós-cirúrgico mão D    30       6     sim 40.11   7
6   SVTO       <NA>                <NA>    NA      NA    <NA> 25.01   2
```

Em merge2, veja que todos os indivíduos de y foram mantidos, porém os valores que não há correspondência com x foram tratados como NA.

```
merge3 <- inner_join(todosFinal, imcDor, by = "id")
merge3
      id       data              reabili idade sessoes termino   imc dor
1   FSTE 04/02/2022   fratura madibular    26       5     sim 24.82   8
2   HESG 09/02/2022       torção joelho    15       7     sim 25.21   9
3   LGSG 14/08/2022                asma    42       4     sim 30.42   2
4   MAJM 30/09/2022  distúrbio cardíaco    68       8     sim 21.29   5
5 NEDGHD 16/04/2022 pós-cirúrgico mão D    30       6     sim 40.11   7
```

Em merge3, veja que ficaram apenas os indivíduos os quais há correspondência entre x e y.

```
merge4 <- full_join(todosFinal, imcDor, by = "id")
merge4
      id       data              reabili idade sessoes termino   imc dor
1    FSTE 04/02/2022   fratura madibular    26       5     sim 24.82   8
```

```
2    HESG 09/02/2022       torção joelho    15       7     sim 25.21   9
3  MCTEAF 19/05/2022           amputação    74       4     nao    NA  NA
4    LGSG 14/08/2022                asma    42       4     sim 30.42   2
5    MAJM 30/09/2022  distúrbio cardíaco    68       8     sim 21.29   5
6    ATFG 07/02/2022                DPOC    54       7     sim    NA  NA
7     THB 08/02/2022           escoliose    26       4     nao    NA  NA
8  NEDGHD 16/04/2022 pós-cirúrgico mão D    30       6     sim 40.11   7
9    PHSW 21/05/2022            COVID-19    41       5     nao    NA  NA
10   KJSF 27/06/2022               neuro    65       9     sim    NA  NA
11   SVTO       <NA>                <NA>    NA      NA    <NA> 25.01   2
```

Veja que aqui estão dos os indivíduos possíveis. Os dados onde não há correspondência foram tratados como NA.

3.10 SALVANDO/CARREGANDO SEU AMBIENTE GLOBAL DE TRABALHO

Sempre que trabalhamos no R e formos fechar a janela após o término de nossas análises, o RStudio me perguntará se eu quero salvar o ambiente global de trabalho. Ou seja, salvar todos os objetos que foram criados e estão sendo utilizados no R (presentes na aba *Environment*). Se isso for do seu interesse, basta clicar em "sim" e escolher uma pasta do seu dispositivo e um nome para o arquivo. Você verá que o R criará um arquivo com a extensão ".R".

Salvar resultados que foram armazenados em objetos talvez seja essencial para que você não perca informação. Principalmente se você trabalhar com o R "cru" (demonstrado no início do capítulo 1 – Figura 1.1), que geralmente é utilizado em servidores. Se a sua necessidade te levou a trabalhar em um terminal e não tem o RStudio e todos os seus botões e mecanismos visuais para interagir com o R, eu vou passar para você duas funções interessantes: uma para salvar esse ambiente global de trabalho e a outra para carregar um ambiente de trabalho que foi salvo anteriormente. Para salvar, basta rodar a função:

```
save.image("caminho_da_pasta/nome_do_arquivo.R")
```

A partir desse momento você pode fechar o R, pois os objetos serão salvos nesse arquivo. Entretanto, devo destacar que pacotes serão fechados juntos com o R. Então, após o carregamento do ambiente global, precisamos rodar novamente todas as bibliotecas (o que não custa nada, já que tudo estará descrito no seu *script* – espero!).

Para carregar ou reabrir o ambiente global de trabalho salvo, basta rodar a função:

```
load("caminho_da_pasta/nome_do_arquivo.R")
```

Capítulo 5

Amostragem

Você já sabe o que é uma amostra? Quando chegamos em uma padaria para tomar um delicioso café, por vezes os funcionários nos oferecem um pequeno pedaço de um alimento sólido ou uma bebida. Isso é para dar validade ao cliente de que aquele alimento é bom e você deveria comprar o inteiro.

Se você parar para perceber, corriqueiramente, as nossas perguntas científicas recaem sobre a população. Por exemplo:

- Quem tem mais sangue circulante no corpo: homens ou mulheres?
- Pessoas sedentárias apresentam valores de triglicerídeos sérico mais elevado do que pacientes atletas?
- A frequência de batimentos cardíacos de gatos é maior do que em cães?

Reconheça que as três perguntas recaem a uma faixa de pessoas ou animais bem abrangente. Ou seja, para saber quem tem mais sangue circulante no corpo, pelo menos em média, eu preciso conferir em todas as pessoas vivas da Terra? E os triglicerídeos, serão dosados em todas as pessoas sedentárias e atletas de todos os países? E quem vai conseguir conferir a frequência de batimentos cardíacos de todos os gatos e cães do mundo?

O primeiro conceito que você precisa conhecer nesse capítulo é o de **população**, que é a soma de todos os indivíduos ou elementos que o estudo tem interesse em obter informação. Pode ser chamado também de universo no âmbito da estatística. Porém, o segundo conceito que você irá aprender é o de **amostra**, que é o subconjunto representativo de uma população.

Coletar dados de uma população pode ser muito difícil, e por vezes, impossível. Por isso, é muito mais comum e factível coletarmos dados de alguns indivíduos dentro daquele universo e assim seguirmos adiante com nossas análises estatísticas para respondermos às perguntas científicas.

Eu quero que você entenda que o conceito de população pode ficar subjetivo, pois se restringimos as características dos indivíduos na pergunta científica, podemos delimitar uma população mais restrita. Vamos a exemplos: desejamos saber se alunos de uma determinada universidade têm elevado risco a desenvolver ansiedade. Perceba que restringimos os indivíduos. Eu não quero saber se toda as pessoas do planeta Terra estão propensas à ansiedade. Nem tão pouco desejamos saber se os brasileiros têm essa propensão. Ainda, não é da nossa importância, nesse momento,

saber se são todos os alunos universitários do Brasil. O meu interesse é em uma universidade específica. Então, é possível que a coleta completa de toda a minha população seja factível.

Por outro lado, na maioria das vezes isso não o é. Se desejarmos saber se pacientes asmáticos atópicos apresentam eosinofilia, meu caro e minha cara leitora, não há a mínima possibilidade de fazer hemograma em todos os pacientes asmáticos que têm no mundo. Por conta disso, fazemos a nossa pesquisa coletando dados de uma quantidade de indivíduos que seja satisfatória para respondermos a esse questionamento.

Há várias questões em jogo as quais coletar dados de uma amostra é mais vantajoso. Primeiro, que populações podem ser extremamente grandiosas, precisamos de tempo hábil para coletar esses dados e recursos financeiros são limitados.

Contudo, preciso reforçar o que foi dito acima: a amostra precisa ser representativa da população. Se quisermos contabilizar e comparar as frequências cardíacas de cães e gatos, muito cuidado se você for verificar essas frequências em uma clínica veterinária, pois é mais provável que você encontre cães e gatos debilitados e estressados. Então as frequências cardíacas talvez não sejam fidedignas a cães e gatos nessas condições. Seria um viés.

Dessa forma, **amostragem** é o procedimento prático de escolha de indivíduos ou elementos de uma população, de forma tal que a população de interesse esteja ali bem representada.

Nem toda estratégia de amostragem é fácil e bem sucedida. Imprudentemente podem ocorrer **erros de amostragem**, quando há desigualdades entre os resultados obtidos de uma amostra com o que realmente apresenta e caracteriza a população de interesse. Na maioria das vezes está envolvida a quantidade de indivíduos na amostra e a seleção enviesada desses. Para reduzir o máximo possível desses vieses, há métodos de amostragem e cálculos para estabelecermos a quantidade mínima aceitável de indivíduos para que essa amostragem seja o mais fiel possível à população.

Existem vários meios para se obter uma amostra. Aqui, vamos abordar simplificadamente três tipos:

- amostra probabilística (ou aleatória, ou casual)
- semiprobabilística
- não probabilística (ou de conveniência)

4.1 AMOSTRA PROBABILÍSTICA

Também chamada comumente de amostra aleatória, é definida por seleção de unidades da população ao acaso. Teoricamente, cada unidade da população tem a mesma chance se ser selecionada para constituir essa amostra.

Exemplificando, imaginemos a população de um determinado município de 100 mil pessoas. Essas pessoas têm seus Cadastro de Pessoa Física (CPF) cadastrado em nos servidores desse município e desejamos fazer uma amostragem, sorteando aleatoriamente 200 desses números. Pronto! Temos aqui uma amostra probabilística. Esse exemplo é clássico tipo de **amostra aleatória simples**, porque todos os indivíduos desse município têm igual chance de serem escolhidos.

Mas, vamos supor que o nosso desejo é sortear 200 pessoas, só que levando em conta a renda mensal, onde abrange: (1) pessoas que ganham menos de um salário mínimo; (2) pessoas que ganham exatamente um salário mínimo; (3) pessoas que ganham até 3 salários; e, (4) pessoas que ganham 4 salários mínimo em diante. O que acabamos de fazer é dividir essa população em **estratos** (exatamente quatro deles). Para que na nossa seleção haja 200 pessoas representativas de cada estrato, podemos fazer um sorteio de 50 unidades em cada estrato. Acabamos de definir uma **amostra aleatória estratificada**.

Criticamente você precisa observar se a sua população em análise é mais homogênea ou heterogênea em diversos aspectos. Se sua decisão recair sobre ser mais homogênea, a amostragem aleatória simples é mais indicada. Contudo, se você verificar que a população é heterogênea, opte pela amostragem aleatória estratificada, mas tenha cuidado em definir os estratos usando uma característica de relevância com a sua pergunta.

Na prática, podemos usar a função sample() do R base para selecionarmos números aleatórios a partir de uma quantidade predeterminada de unidades de um universo. Por exemplo, vamos supor que a nossa população tenha 60 unidades. Desejamos fazer uma amostragem aleatória simples selecionando ao acaso 6 números desse universo. Procedemos o seguinte comando:

```
sample(x = 60, size = 6)
[1] 20 26 55 46 22 32
```

Onde “x” é o parâmetro que indicamos o tamanho da nossa população; “size” é a quantidade amostral que desejamos retirar desse universo.

Não sei se você percebeu, caro leitor ou leitora, mas acabamos de selecionar aleatoriamente seis números para jogarmos na Mega Sena. Se você resolver jogar a partir do que eu te ensinei, lembre-se de mim quando ganhar o prémio.

No caso acima, por padrão, os valores retornados pelo R não se repetirão, porque há um parâmetroo chamado "replace" que por padrão é definido como "FALSE". A maioria das amostragens são corriqueiramente realizadas dessa forma, porém, há situações em que podemos escolher uma unidade mais de uma vez, sem prejuízo à pesquisa. Dessa forma, perceba a diferença nessas duas situações:

```
sample(7, 6, replace = FALSE)       # x = 7 e size = 6
[1] 4 2 5 7 3 6

sample(7, 6, replace = TRUE)
[1] 6 5 1 3 6 1
```

Perceba que no primeiro caso os algarismos não se repetem, enquanto que no segundo código houve a repetição do "6" e do "1". Essa reposição é aleatória e, portanto, é possível que haja seleção de números totalmente diferentes usando "replace = TRUE".

Para exemplificar na prática uma amostra aleatória estratificada no R, vamos recorrer ao banco de dados "iris". Nele, há três estratos, que são os três tipos de espécies tabuladas. Certamente há diversas maneiras de fazermos essa amostragem, mas gostaria de mostrar a você um método com o passo a passo para que você consiga compreender cada etapa do processo:

- Primeiro passo = filtrar os estratos em sub-bancos;
- Segundo passo = realizar a amostragem em cada sub-banco;
- Terceiro passo = compilar a amostragem em um banco final e definitivo.

Vamos fazer uma amostragem retirando ao acaso dez observações de cada estrato, tendo por fim uma amostra constituída de 30 unidades. A função nrow() a seguir retornará o número de observações em cada sub-banco, que será o nosso subuniverso temporário. Na prática:

```
data(iris)  # abrir o banco iris
library(tidyverse) # abrindo a biblioteca tidyverse

#Criação dos sub-bancos
df1 <- iris %>% filter(Species == 'setosa')
df2 <- iris %>% filter(Species == 'versicolor')
df3 <- iris %>% filter(Species == 'virginica')

#Realizando a amostragem em cada sub-banco
amostragem1 <- sample(nrow(df1), 10)
amostragem2 <- sample(nrow(df2), 10)
amostragem3 <- sample(nrow(df3), 10)

#Definindo o banco final
dfFinal <- rbind(
  df1[amostragem1, ],
```

```
  df2[amostragem2, ],
  df3[amostragem3, ]
  )

#visualize dfFinal
View(dfFinal)
```

```
# Confirmando o número de linhas e colunas do data frame final
nrow(dfFinal)
[1] 30

ncol(dfFinal)
[1] 5
```

O banco original "iris" apresenta cinco variáveis (colunas). Nosso *data frame* final, resultante da nossa amostragem aleatória estratificada, aparentemente está correto, pois apresenta 30 linhas e as 5 variáveis.

4.2 AMOSTRA SEMIPROBABILÍSTICA

Quando a amostra é retirada de maneira limitadamente aleatória, ou seja, houve um processo o qual não podemos dizer que foi completamente aleatória, aí sim estamos diante de uma amostragem semiprobabilística. Existem três tipos básicos de amostras semiprobabilísticas, com seus respectivos processos de amostragem: amostra por conglomerados, sistemática e por quotas.

4.2.1 Amostra por conglomerados

Imagine que ao invés de selecionarmos indivíduo por indivíduo, selecionaremos um certo conjunto de indivíduos que estão agrupados por algum motivo. Por exemplo, desejamos estudar pacientes cardiopatas no estado do Maranhão, só que ao invés de irmos em todos os hospitais do estado, vamos sortear aleatoriamente uma quantidade x de hospitais. Cada hospital aqui é um agrupamento, que conterá os indivíduos que serão estudados.

Vamos praticar e entender melhor isso no R. Exemplo: a Secretaria de Saúde de um determinado município deseja averiguar e acompanhar os jovens na faixa etária de 9 a 14 anos que tomaram ou não vacina contra o HPV (papilomavírus humano). Como precisa-se de muito recurso para o acompanhamento, percebeu-se que seria melhor escolher 5 das 7 escolas totais que o município tem em funcionamento com esse público. Vamos listar as escolas e das uma identificação de 1 a 7 a cada uma delas e assim criamos o seguinte *data frame*:

```
#amostragem semiprobabilística - conglomerados
df <- data.frame(
  id = seq(1:7),
  escola = c('Maria Quitéria', 'Dom Manuel', 'Dois de Julho', 'Dom Pedro II', 'Ma-
chado de Assis', 'Central', 'Anita Malfatti')
)
```

Para escolhermos aleatoriamente cinco escolas, basta rodar o comando:

```
sample(7, 5)
[1] 1 5 3 4 2                  #ordem das escolas selecionadas
```

Nesse exemplo, as escolas Municipais Maria Quitéria, Machado de Assis, Dois de Julho, Dom Pedro II e Dom Manuel foram selecionadas. Dessa forma, os pré-adolescentes de 9 a 14 anos dessas escolas serão a representatividade de todos os pré-adolescentes de 9 a 14 anos desse município, neste caso.

4.2.2 Amostra sistemática

Uma amostragem sistemática necessita de uma lista ordenada, como a variável "id" que criamos acima. Esse método visa retirar da população uma quantidade k de unidades, sendo que a primeira é retirada ao acaso e as demais serão retiradas por um sistema.

Vamos supor que desejamos retirar 200 unidades de uma população de 4.000 indivíduos. Se fizermos a divisão 4000:200 = 20 (esse é o nosso sistema aqui; de vinte em vinte), ou seja, após a seleção aleatória do primeiro indivíduo, incluiremos elementos na nossa amostragem de 20 em 20 indivíduos.

```
#amostragem semiprobabilística - sistemática
sample(20, 1)  #sorteando o primeiro elemento
[1] 16        (o primeiro elemento sorteado foi o 16°)

#listando sistematicamente os elementos de 20 em 20, começando no 16
lista <- seq(16,4000, 20)

#visualizar todos selecionados
lista
[1]   16   36   56   76   96  116  136  156 ...
```

O comando acima é um método muito simples de realizar esse tipo de amostragem. Mas, lembre-se de mudar os argumentos das funções sample() e seq() de acordo com a sua pesquisa e necessidade. Para sample(), coloque primeiro o valor do sistema e depois o valor 1 (retirar uma unidade desse universo). Para seq(): (1) inclua o valor do primeiro colocado achado na fase da função sample(); (2) depois, o valor do último elemento; (3) por fim, o sistema.

Você conseguiu compreender que uma variável numérica que ordene os indivíduos é de grande importância? Me refiro a variável "id" que criamos em diversos exemplos neste e em outros capítulos. Com ela nós podemos não só identificar cada unidade da amostra, mas podemos usá-la para selecionarmos indivíduos segundo uma subamostragem.

4.2.3 Amostra por quotas

Esse tipo de amostragem é muito parecido com a amostragem aleatória estratificada, mas de aleatoriedade tem-se muito pouco. Aqui, não escolhemos ao acaso um indivíduo pertencente a um grupo, mas, determinamos certas características de algum tipo de grupo e iremos procurar pessoas/animais/plantas com essas características. Perceba que a seleção da amostra é feita por julgamento prévio, que depois confirmaremos tais características entre cada unidade.

Esse tipo de amostragem é muito utilizado em pesquisa de mercado e levantamento de opinião. Exemplificando, vamos supor que um determinado grupo de mercado deseja saber a preferência de usuários dentre dois laboratórios, para planejamento de futuras parcerias comerciais. Para ser uma amostragem por quotas, o pesquisador quer entrevistar adultos maiores que 45 anos. Então, os entrevistadores sairão à procura de pessoas daquela localidade que visualmente pareçam com idade superior a 45 anos. Ao abordar um entrevistado, iremos saber se ele corresponde mesmo a essa característica ou não.

4.3 AMOSTRA NÃO PROBABILÍSTICA OU DE CONVENIÊNCIA

Uma amostra por conveniência, também chamada de acidental, é totalmente não probabilística, pois o pesquisador escolhe as unidades pela facilidade de acesso a elas, por essas estarem em um determinado local ou tempo disponíveis.

Deve-se ter muito cuidado com esse tipo de amostragem, pois, apesar da facilidade de acesso, é muito fácil acharmos erros de tendência, que podemos chamar de viés.

Vamos supor que um médico residente, em sua pesquisa de especialização, se questione quais as comorbidades mais frequentes em pacientes de um hospital universitário, especializado em cardiologia. A conveniência aqui é que serão recrutados pacientes do mesmo hospital que ele trabalha, porém, há um certo erro embutido nessa pesquisa. Temos algumas questões que serão complicadas para responder, por exemplo: nesse hospital há indivíduos de que faixa etária? Qual

a classe social dessas pessoas? Em sua maioria são pacientes graves? As principais doenças cardíacas desses pacientes são as mesmas da população em geral?

É bom que se saiba que não estamos tentando invalidar a pesquisa (até mesmo porque ela tem a sua importância), mas devemos estar atentos à estrapolação desses achados para a população como um todo. Se por um acaso esse residente encontrar alta frequência de arritmias, isso não quer dizer exatamente que arritmia é o problema cardíaco que mais aflige a população de cardiopatas do Brasil, mas sim que é a cardiopatia mais prevalente daquele hospital.

A chave aqui é: se a pesquisa trata seu estudo com amostra por conveniência, tenha cuidado com as palavras escritas no texto científico. Conflite criticamente esses achados com o que já foi descrito na literatura.

Para ter uma pesquisa satisfatória, que responda a minhas perguntas biológicas, qual a quantidade mínima necessária de indivíduos/unidades eu preciso ter para fazer posteriormente as análises estatísticas? Essa é uma pergunta fundamental e inicial a qualquer pesquisa a ser feita. Tentaremos responder em capítulos à frente onde falaremos de poder de teste e tamanho amostral.

Capítulo 5

Medidas de Tendência Central

Por vezes, até intuitivamente, precisamos de um número que resuma todo um conjunto de dados. Qual a média de arrobas de bois para abate tem um determinado produtor? Sabendo esse número o produtor pode tomar decisões: vender agora ou esperar um pouco mais o período de engorda.

Esse número é na realidade uma medida de centro, que resume todos os pesos dos animais que vão variar entre esse número. Média talvez seja a mais conhecida, contudo, temos também a mediana e a moda. Vamos conversar um pouco sobre cada uma delas.

5.1 MÉDIA ARITMÉTICA

Na matemática, existem alguns tipos de médias, como: a média aritmética simples, média aritmética ponderada, a média geométrica e a média harmônica. Nesse livro abordaremos a média aritmética simples, que é a mais utilizada, a ponto de resumidamente chamarmos meramente de média.

Para calcularmos a média de uma distribuição de dados, basta somarmos todos os valores e dividirmos o resultado pela quantidade de dados. Matematicamente:

$$\bar{x} = \frac{\Sigma x}{n}$$

Como estamos calculando a média de uma amostra, usamos o símbolo $\bar{x}$ (lê-se x-barra, ou ainda x-traço). Quando calculamos a média de uma população, a partir de um censo, usa-se a letra grega **μ**. O desejável é que $\bar{x}$ seja o mais próximo possível de μ. Já o símbolo Σ indica soma dos elementos, que no nosso caso serão todos os valores de x. Dividiremos pelo total, que aqui está representado por n.

Para exemplificar, vamos supor que uma diretora de uma creche esteja preocupada com o peso das crianças. Mediu o peso de sete que estudavam em uma determinada sala. Qual a média de peso dessas crianças? Vejamos.

```
x <- c(8.9, 9.1, 7.3, 8.4, 9.0, 7.6, 8.3) #peso de sete crianças em kg
media <- sum(x) / 7
media
[1] 8.371429
```

Então, podemos dizer que a média de peso das crianças é de 8,37 kg. Perceba que usamos a função sum(), que irá somar todos os valores de x. Abaixo, vamos incrementar nosso código com

a função length(), que retorna o comprimento de um objeto. Veja que o valor retornado é exatamente o mesmo.

```
sum(x) / length(x)
[1] 8.371429
```

Para a nossa felicidade, o R já traz uma função para calcularmos a média de uma distribuição diretamente, é a função mean():

```
mean(x)
[1] 8.371429
```

Vamos exemplificar mais uma vez, agora usando o banco de dados chamado CO2, presente no R. Esses dados vêm de uma pesquisa que quantificou a captação de gás carbônico pela espécie *Echinochloa crus-galli*. A absorção de CO_2 foi dada em µmol/m² seg. É importante sempre expor o resultado das medidas de tendência central com a sua unidade respectiva. Deixar de mostrar a unidade é um erro, diria até ingênuo. Evite cometê-lo.

Vamos abrir o banco de dados e calcular a média de absorção que está na variável "uptake".

```
#abrindo o banco
data(CO2)

#calculando a média
mean(CO2$uptake)
[1] 27.2131
```

Mesmo que:

```
sum(CO2$uptake)/length(CO2$uptake)
[1] 27.2131
```

Em suma, a média de CO_2 absorvido pelas plantas naquele momento da pesquisa foi de aproximadamente 27,21 µmol/m² seg.

Se você não especificar ao R qual a variável de interesse, ele retornará um erro:

```
mean(CO2)
[1] NA
Warning message:
In mean.default(CO2) : argumento não é numérico nem lógico: retornando NA
```

5.2 MEDIANA

A mediana é a medida que ocupa exatamente a posição central de um conjunto de dados que esteja ordenado. Dessa forma, a mediana acaba dividindo a distribuição de dados em duas partes. Por exemplo:

{5, 6, 8, 9, 11}

A mediana desse conjunto de dados é o número 8, porque é o valor que está exatamente no meio, com os **dados ordenados**. Muito simples, não? Veja que o n amostral, ou seja, a quantidade de observações desse conjunto de dados é ímpar; igual a cinco observações.

Mas, e se a quantidade de observações for par?

{17, 19, 21, 22}

Nesse caso, fazemos a média aritmética dos dois números centrais. Nesse exemplo, a mediana será (19 + 21) / 2 = 20.

Como a maioria dos bancos de dados são extensos e ficar ordenando e procurando a mediana pode custar um tempo, o R tem uma função própria que retorna a mediana para gente: a median(). Veja:

```
x1 <- c(5, 6, 8, 9, 11)
median(x1)
[1] 8

x2 <- c(17, 19, 21, 22)
median(x2)
[1] 20
```

Usando um banco mais abrangente, podemos voltar para o CO2. Vamos verificar qual é a mediana de gás carbônico absorvido pelas plantas:

```
median(CO2$uptake)
[1] 28.3
```

Dessa forma, a mediana é 28,3 µmol/m² seg. Se você desejar observar os dados de forma ordenada, para a partir disso buscar a mediana, use a função sort() do R:

```
sort(CO2$uptake)
 [1]  7.7  9.3 10.5 10.6 10.6 11.3 11.4 12.0 12.3 12.5 13.0 13.6 13.7 14.2 14.4 14.9
15.1 16.0 16.2 17.9 17.9 17.9 18.0 18.1 18.9 18.9
[27] 19.2 19.4 19.5 19.9 21.0 21.9 22.0 22.2 24.1 25.8 26.2 27.3 27.3 27.8 27.9 28.1
28.5 30.0 30.3 30.4 30.6 30.9 31.1 31.5 31.8 32.4
[53] 32.4 32.4 32.5 34.0 34.6 34.8 35.0 35.3 35.4 35.5 37.1 37.2 37.5 38.1 38.6 38.7
38.8 38.9 39.2 39.6 39.7 40.3 40.6 41.4 41.4 41.8
[79] 42.1 42.4 42.9 43.9 44.3 45.5
```

Perceba que o n amostral é par (84), então, o R teve calcular a média de dois valores centrais (28,1 e 28,5) para obter a mediana:

```
length(CO2$uptake)
[1] 84

mean(c(28.1,28.5))        #a média dos valores centrais é a mediana
[1] 28.3
```

5.3 MODA

A moda é o valor numérico que mais se repete dentro de uma distribuição de dados, ou seja, é o valor com maior frequência.

Aqui, não precisa ordenação de dados, mas apenas buscar o que mais se repete. Veja:

{3, 6, 2, 3, 4, 9, 15, 3, 2, 6, 3, 1}

O valor 3 é o que tem maior frequência, pois ele se repetiu quatro vezes. Então, 3 é a moda.

No R, até o presente momento não há uma função nativa que retorne a moda de um conjunto de dados, mas podemos construir facilmente. Abaixo, estão duas funções criadas que retornarão a moda:

```
dados <- c(3, 6, 2, 3, 4, 9, 15, 3, 2, 6, 3, 1)

moda1 <- function(x){
   uniqx <- unique(x)
   uniqx[which.max(tabulate(match(x, uniqx)))]
 }

moda2 <- function(x){
   modax <- table(x)
   names(modax)[modax == max(modax)]
 }

m1 <- moda1(dados)       #retorna a moda usando a função moda1
m2 <- moda2(dados)       #retorna a moda usando a função moda2

m1
[1] 3

m2
[1] "3"
```

Percebeu que as duas funções conseguiram retornar corretamente a moda? Porém, o formato retornado por m1 parece diferente do formato retornado por m2. Vamos verificar a classe do objeto que cada uma das funções retornou:

```
class(m1)
[1] "numeric"

class(m2)
[1] "character"
```

Então, a função moda1, do jeito que foi construída, retornará um número quando o banco de dados é numérico. Já a função moda2, retornará na forma de caractere, ou seja, um dado categórico. Qual é o melhor? Aquele da sua necessidade.

Como dito por Triola (2017), a moda é a única medida de tendência central que pode ser usada com dados nominais. Dessa forma, essas funções podem retornar a moda para variáveis categóricas de maior frequência também. Por exemplo, supondo um banco fictício de acidentes atendidos por uma ambulância da SAMU (Serviço de Atendimento Móvel de Urgência):

```
dados <- c('carro', 'moto', 'doméstico', 'moto', 'elétrico',
           'arma de fogo', 'faca', 'moto', 'moto', 'arma de fogo')

moda1(dados)
[1] "moto"

moda2(dados)
[1] "moto"
```

No caso acima, a moda foi os atendimentos de acidentes por moto (motocicleta). Como há apenas uma moda, uma frequência máxima, dizemos que essa é uma moda é unimodal.

As modas podem ser classificadas de acordo com o surgimento quantitativo de modas, ou seja:

- Unimodal: quando existir apenas um valor com maior frequência;
- Bimodal: quando existirem dois valores com a mesma frequência máxima, ou seja, tem duas modas;
- Trimodal: quando existirem três valores com a mesma frequência máxima ou três modas;
- Multimodal: quando mais de três valores se repetirem com a mesma frequência máxima.
- Amodal: quando não possuir nenhum valor repetido (não há uma moda).

Vamos demonstrar um exemplo de uma moda categórica trimodal bastante conhecida no R, usando os dados do banco "iris". Já é sabido que os pesquisadores trabalharam com três espécies diferentes, e que coletaram 50 unidades de cada uma. Veja:

```
table(iris$Species)

    setosa versicolor  virginica
        50         50         50
```

```
moda1(iris$Species)      #verificando através da função “moda”
[1] setosa
Levels: setosa versicolor virginica

moda2(iris$Species)   #verificando através da função “moda2”
[1] "setosa"     "versicolor" "virginica"
```

5.4 RELAÇÕES ENTRE A MÉDIA, MEDIANA E A MODA

Em capítulos à frente falaremos melhor sobre distribuição de dados. O mais simples que posso explicar no momento é a distribuição normal, aquela cujo gráfico chamado histograma aparece com uma tendência de sino. Veja na Figura 5.1 que o eixo x revela a dimensão dos dados, ou seja, os valores vão -20 a 120. O eixo y revela a frequência de cada valor, então, perceba que nesse banco de dados há muitos números perto do valor 50. O código para gerar esse gráfico está logo abaixo:

```
#Figura 5.1
set.seed(1234)
Normal <- round(rnorm(2000, mean = 50, sd = 20),digits = 0)
hist(Normal, freq = F, breaks = 20, main = 'Distribuição normal')
curve(dnorm(x, mean = mean(Normal), sd = sd(Normal)),
      from = -50, to = 150, col = 'red', lwd = 2, add = T)
```

Estamos criando um objeto chamado “Normal” o qual escolherá aleatoriamente valores que estarão na distribuição normal, usando a função rnorm(). Reduzimos os decimais dos números a zero usando a função round(), com seu parâmetro “digits” = 0. Após, construímos o histograma (o gráfico com barras) com a função hist(). A curva de tendência foi construída com a função curve().

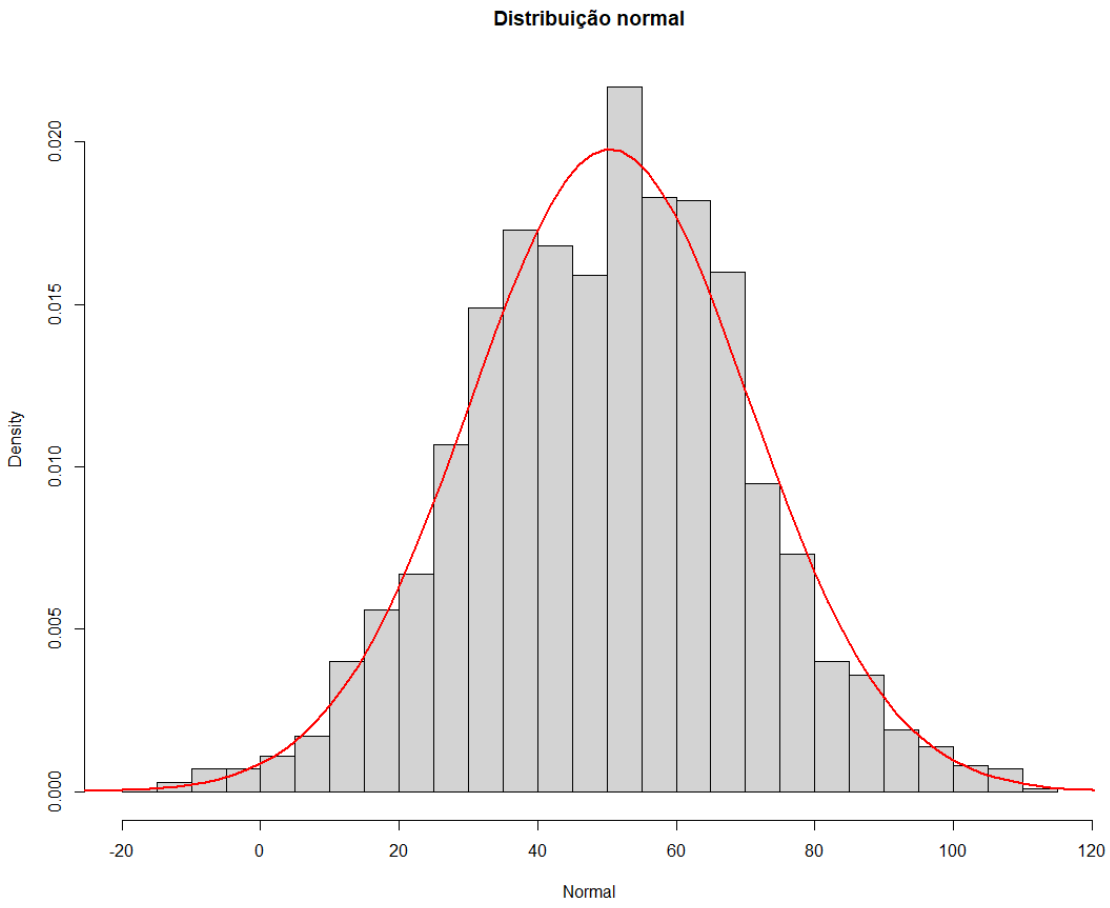


Figura 5.1. Dados em uma distribuição normal.

O objetivo principal aqui é observarmos onde estão a média, a mediana e a moda nesse gráfico. Para calcularmos, vamos usar os comandos:

```
mean(Normal)
[1] 50.4605

median(Normal)
[1] 51

moda1(Normal)
[1] 51
```

Lembrando que a função `moda1()` foi construída por nós mais acima, podemos perceber que os valores entre a média, mediana e a moda são bem próximos (inclusive, mediana e moda têm o mesmo valor). Vamos colocar no gráfico esses valores em forma de linhas coloridas para identificarmos melhor:

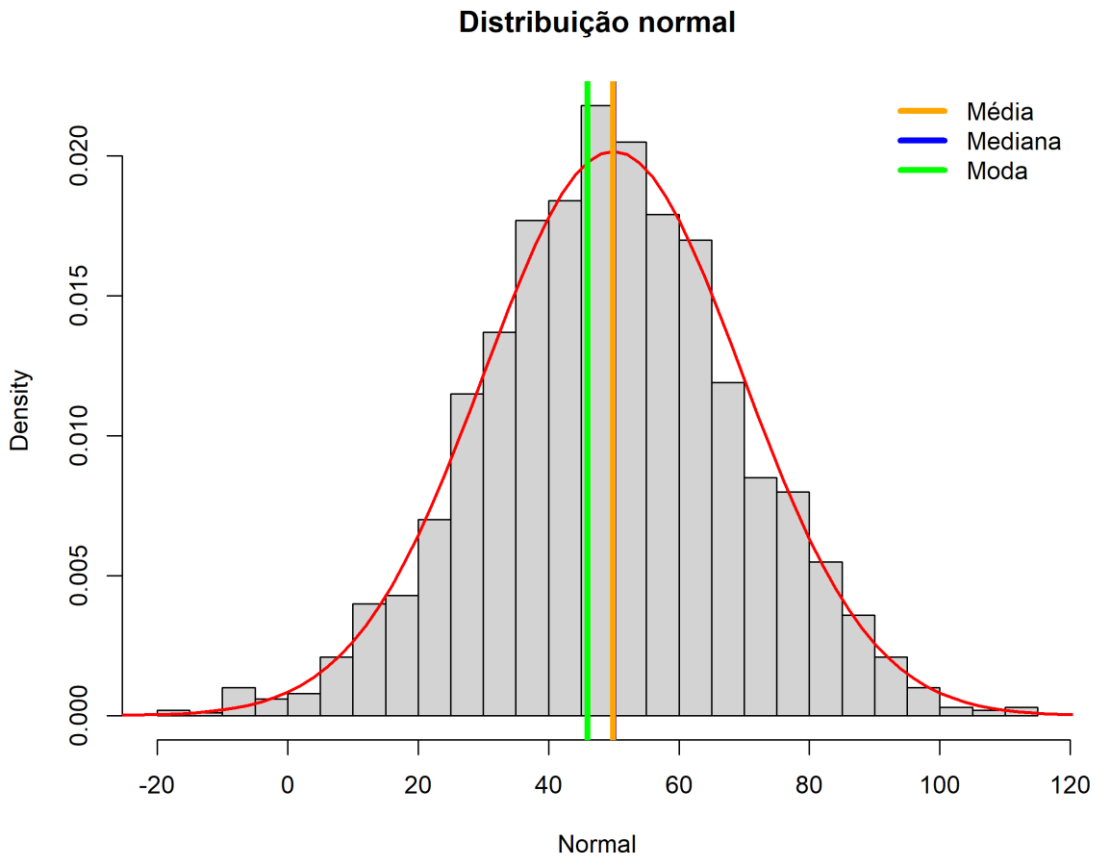


Figura 5.2. Distribuição normal de dados com a média, mediana e moda definidas.

Analisando a Figura 5.2, verificamos que as linhas verticais que definem média, mediana e moda são realmente muito próximas. Inclusive, a linha da mediana se sobrepôs à moda, por terem valores iguais.

Entretanto, quando a distribuição dos dados não está na normal, podemos encontrar o arranjo dos dados da forma mostrada na Figura 5.3, por exemplo. Perceba que a tendência do sino já não é mais verdadeira, pois o histograma tende mais para um lado do gráfico do que para outro. Vejamos os códigos:

```
#Figura 5.3
set.seed(123)
foraNormal <- rchisq(50000, df = 5)
hist(foraNormal, main = 'Fora da distribuição normal',
     freq = FALSE,
     xlim = c(0,30),
     ylim = c(0,0.2))
curve(dchisq(x, df = 5), from = 0, to = 30,
      n = 5000, col= 'red', lwd=2, add = T)
```

Estamos construindo um objeto chamado "foraNormal", cuja função usada foi rchisq(), utilizada para criar distribuição de qui-quadrado (que nos aprofundaremos em capítulos à frente). Com a função hist() teremos o histograma e com a função curve() fizemos a linha de tendência da distribuição.

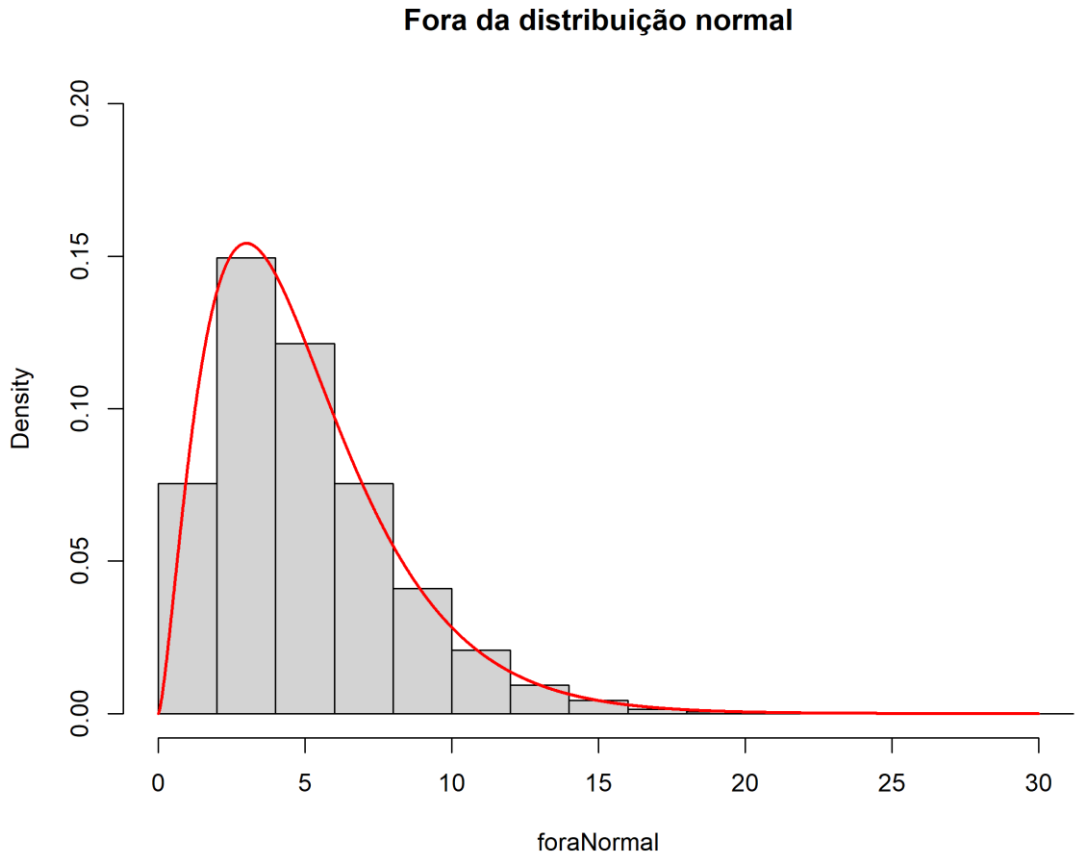


Figura 5.3. Distribuição fora da normal.

Primeiramente precisamos observar que esses dados têm uma amplitude diferente dos dados exemplificados mais acima. Aqui, o valor máximo está próximo de 30. Para sabermos os valores da média, mediana e moda, basta rodarmos os comandos:

```
mean(foraNormal)
[1] 4.996948

median(foraNormal)
[1] 4.362124

moda1(foraNormal)
[1] 2.571802
```

Podemos ver esses parâmetros estatísticos na Figura 5.4, usando os comandos:

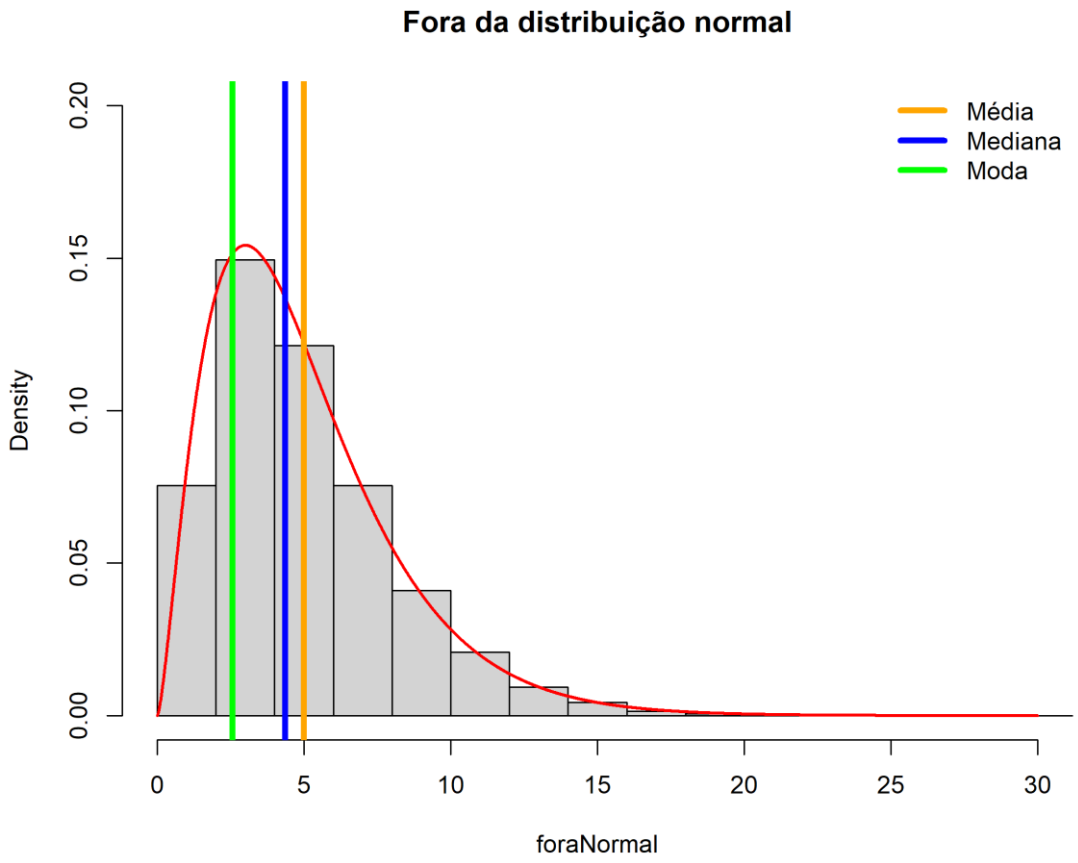


Figura 5.4. Distribuição fora da normal, com a média, mediana e moda definidas.

Perceba que os valores estão até próximos, mas não chegam a estar sobrepostos. A questão que devemos nos debruçar aqui é: entre qual das três medidas de tendência central melhor representa essa distribuição? A média? A mediana? A moda?

Se você perceber, a média é muito influenciada pelos valores extremos, uma vez que para contabilizá-la precisamos somar todos os valores. Isso não acontece com a mediana, nem com a moda. Entretanto, a moda é mais grosseira, se baseando meramente no valor mais repetido. Portanto, nos casos em que os dados não estiverem na distribuição normal, damos preferência para a mediana para representar esse conjunto de dados.

Capítulo 9

Medidas de Posição Relativa

As medidas de posição relativa são valores em localizações relativas a outros valores dentro do mesmo conjunto de dados. Aqui, abordaremos os percentis, os quartis e os diagramas de caixa. É importante que se diga desde já que para avaliarmos esses valores, os dados precisam obrigatoriamente estar ordenados.

6.1 PERCENTIS

Imagina comigo o seguinte passo a passo: (1) peguemos um conjunto de dados numéricos grande (mais de 1.000 observações, por exemplo); (2) vamos ordenar esses dados em ordem crescente; (3) agora, vamos dividir esses dados em 100 partes (100 grupos de dados que estavam ordenados). Pois bem, onde a sua cabeça cortou esses dados, são os nossos percentis. Gosto de brincar dizendo que os percentis são as tesouras que contam e separam dados ordenados.

Dessa forma, percentis são valores que dividem um conjunto de dados em 100 partes diferentes, com 10% dos dados em cada grupo. Os percentis são dados em P_1, P_2, P_3, ..., P_{99}. Tome a reta abaixo como a sequência ordenada dos dados de uma distribuição. Perceba onde estão os percentis:

P_1 P_2 P_3 P_4 (...) P_{96} P_{97} P_{98} P_{99}

-------|-------|-------|-------|------- (...) -------|-------|-------|-------|-------

Se você reparar, o P_{50} é o percentil que divide os dados justamente no meio. Que medida de tendencia central divide os dados ordenados ao meio? Isso, a mediana. Pois bem, a mediana é justamente o 50° percentil.

Para acharmos o valor de um percentil qualquer, podemos utilizar a função quantile(). Nela, devemos preencher/atribuir minimamente dois parâmetros: "x" que serão os dados em si; e "probs", que será o percentil desejado na forma de decimal (por exemplo, P_{30} será 0.30; P_{48} será 0.48, e assim por diante). Vamos praticar isso no R usando o banco de dados USArrests, que trata de taxas de crimes violentos em estados dos Estados Unidos. Verifiquemos alguns percentis sobre taxas de assaltos (no banco se apresenta como *Assault*), além de valor mínimo, mediana e valor máximo. Vamos ver se tem sentido?

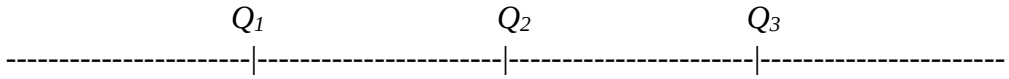
```
data("USArrests")

min(USArrests$Assault)
[1] 45
```

```
quantile(USArrests$Assault, probs = 0.10)   #P10
 10%
56.9

quantile(USArrests$Assault, probs = 0.48)   #P48
   48%
157.56

quantile(USArrests$Assault, probs = 0.50)   #P50
50%
159

median(USArrests$Assault)
[1] 159

quantile(USArrests$Assault, probs = 0.90)   #P90
  90%
279.6

quantile(USArrests$Assault, probs = 0.99)   #P99
   99%
336.02

max(USArrests$Assault)
[1] 337
```

6.2 QUARTIS

Se os percentis dividem a distribuição de dados em 100 diferentes grupos, os quartis dividem os dados em quatro grupos. Simples desse jeito. Então, os quartis nada mais são do que medidas de localização que dividem um conjunto de dados em quatro partes, tendo cada parte cerca de 25% dos dados.

Existem três: Q_1 chama-se de primeiro quartil; Q_2 de segundo quartil e Q_3 é o terceiro quartil. Enxergando didaticamente um conjunto de dados ordenados, vamos ver onde ser localizariam os quartis:



Temos aqui uma coisa interessante a ser mostrada: o segundo quartil é justamente a mediana, que também é o 50° percentil. Então, $Q_2 = P_{50}$ = mediana. Por analogia, Q1 e Q3 também correspondem a percentis, especificamente P_{25} e P_{75}, respectivamente. Dessa forma, fica fácil descobrirmos os quartis no R através da função quantile() já conhecida. Além dessa, existe uma outra maneira bastante utilizada para esse fim, usando a função summary(). Vejamos:

```
#quartis
> quantile(USArrests$Assault, probs = 0.25)   #Q1 = P25
25%
109
> quantile(USArrests$Assault, probs = 0.50)   #Q2 = P50 = mediana
50%
159
> quantile(USArrests$Assault, probs = 0.75)   #Q3 = P75
75%
249

> summary(USArrests$Assault)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   45.0   109.0   159.0   170.8   249.0   337.0
```

O interessante da função summary() é que ela retorna várias medidas com um só comando (valor mínimo, Q_1, mediana, média, Q_3 e valor máximo). Por isso, é largamente usada. Outra coisa que você deve saber é que os quartis também são mais frequentemente analisados do que os percentis.

6.3 DIAGRAMA DE CAIXA OU *BOXPLOT*

Já imaginou visualizar em um gráfico tudo que foi abordado no subcapítulo acima sobre quartis? Não? Pois você perceberá que há um gráfico, chamado de diagrama de caixa ou *boxplot* que resume em imagem todos os quartis. Por conta disso, ele é vastamente explorado no meio acadêmico.

No *boxplot* estão presentes os quartis e valores máximo e mínimo. Por vezes, esses valores máximo e mínimo são muito discrepantes do conjunto de dados geral e se for assim, achamos esses valores de *outliers* (veja Figura 6.1). Exemplo, se estamos aferindo a pressão sistólica entre pessoas normais de saúde, esperasse que esses valores estejam entre 110 a 130 mmHg. Caso encontremos um valor de 300 mmHg, isso certamente é um *outlier*. Não é uma pressão sistólica ventricular de um paciente normal.

Um fato importante é que dentro da caixa há 50% dos dados, ou seja, metade da distribuição está dentro do retângulo. Isso pode ser útil para entender como os dados estão distribuídos.

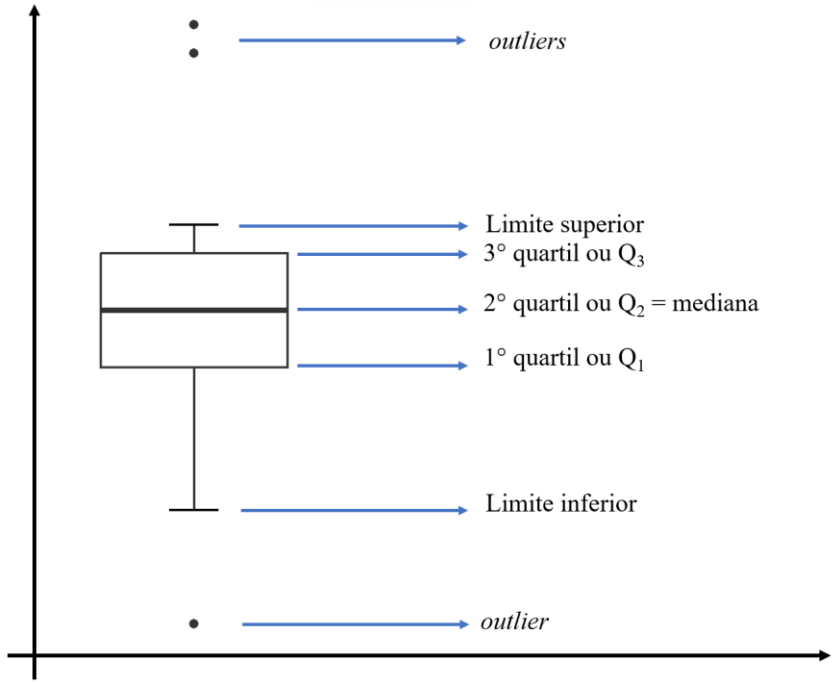


Figura 6.1. Esquema de montagem de um diagrama de caixa (*boxplot*).

Se houver *outliers*, os limites do *boxplot* devem ser calculados segundo uma certa variação dos dados, que veremos a seguir. Caso não haja outliers, os valores mínimo e máximo serão os limites.

Essa tal variação é dada pela amplitude interquartil (AIQ), onde, para calculá-la basta subtrairmos o terceiro quartil do primeiro quartil. Aprofundaremos melhor sobre em capítulos à frente, mas verifique que a AIQ acaba sendo a altura da caixa, segundo o eixo y do gráfico. Outra coisa: quanto maior é essa altura, mais variação há na distribuição dos dados. Se a caixa é pequena, a variação dos dados também é pequena.

$$AIQ = Q_3 - Q_1$$

Para o cálculo dos limites inferior e superior, podemos seguir o *boxplot* de Tukey, o qual é padrão de muitos *softwares* de estatística. Para isso, calculemos os limites da seguinte forma:

$$\text{Limite inferior} = Q_1 - (1{,}5 \times AIQ)$$

$$\text{Limite superior} = Q_3 + (1{,}5 \times AIQ)$$

Na prática, criar um *boxplot* no R é muito simples, usando a função chamada boxplot(). Vamos exemplificar usando dois bancos já abordados: USArrests e o iris.

```
#Figura 6.2
data("USArrests")

#boxplot com uma variável
boxplot(USArrests$Assault)
```

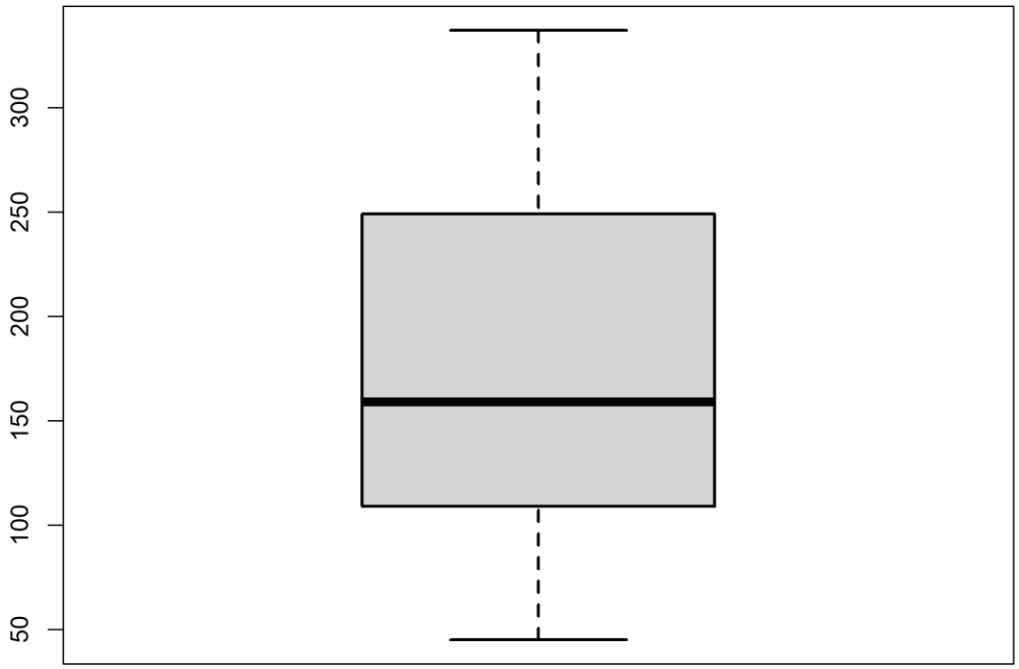


Figura 6.2. Um único *boxplot*.

```
#Figura 6.3
#boxplot com duas ou mais variáveis de um mesmo banco
#(basta lista-las separando por vírgula)
boxplot(USArrests$Assault, USArrests$Murder, USArrests$UrbanPop, USArrests$Rape)
```

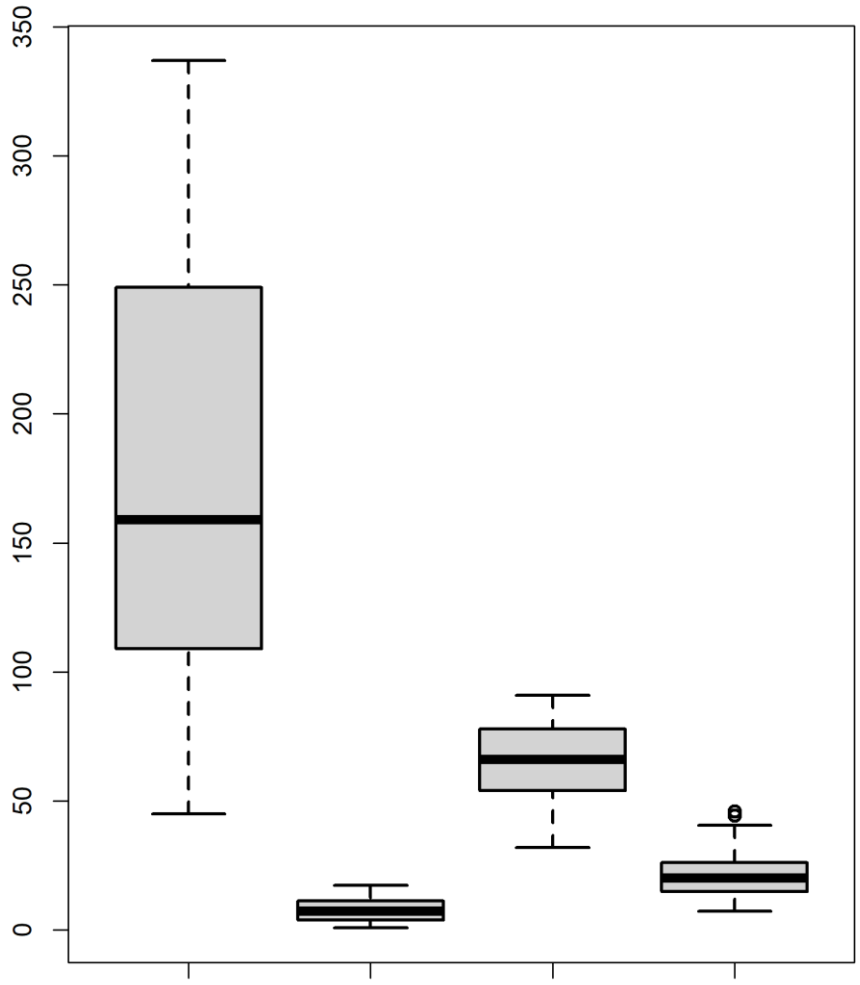


Figura 6.3. *Boxplots* de grupos diferentes.

Por ora, vamos deixar esses gráficos lindos dessa maneira, mas informando a você, leitor e leitora, que dá para melhorar bastante eles. Vamos nos reservar a explorar mais os gráficos usando o pacote ggplot2 em capítulos subsequentes.

Capítulo

7

Medidas de Dispersão

Vamos supor que queiramos medir o peso das crianças entre 1 a 2 anos em duas creches diferentes (A e B). Na creche A nós conseguimos aferir o peso de todas as crianças e a média de peso foi 7,4kg. Na creche B só foi possível fazer a pesagem de 3 crianças e a média entre elas foi de 10,5 kg.

Entre a creche A e B, há crianças muito parecidas: mesmo país, memos estado, mesma cidade, até do mesmo bairro. Eu não sei você, mas em mim há um desconforto em acreditar que a média real de peso das crianças dessa faixa etária seja a média estipulada pela creche B. Foram poucos indivíduos selecionados para essa tarefa e talvez, por algum motivo aleatório, os pesos das crianças sejam tão diferentes que a média esteja superestimada, nesse caso.

O ideal é que os valores não sejam muito diferentes entre si. Quando as análises experimentais e a coleta de dados são bem feitas, os valores obtidos de uma mesma condição tendem a serem bem próximos. Quando isso não é possível, pode ocorrer alta variação entre os dados.

Não sei se você percebeu, mas "contabilizar" a variação dos dados tem importância. Por conta disso, vamos discutir sobre amplitude, intervalo interquartil, variância, desvio padrão e coeficiente de variação. Todos esses parâmetros avaliam quanto os dados divergem entre si.

7.1 AMPLITUDE

A amplitude talvez seja a medida mais simples para avaliarmos a diversidade entre os dados. Para calculá-la, basta subtrairmos o maior valor do menor valor entre os dados.

Damos o nome de máximo para o valor mais elevado e podemos busca-lo utilizando a função `max()` no R. Para buscarmos o menor valor, podemos usar a função `min()` do R base.

$$\text{Amplitude} = \text{máximo} - \text{mínimo}$$

Na prática, podemos calcular a amplitude diminuindo `max()` por `min()`. Para isso vamos criar um banco fictício:

```
dados <- c(12, 14, 15, 11, 17, 14, 16, 13, 13, 12)

min(dados)
[1] 11

max(dados)
[1] 17
```

```
amplitude <- max(dados) - min(dados)
amplitude
[1] 6
```

O exemplo acima é o mais simples que possamos ver, uma vez que os dados contínuos estão em um só vetor e esses correspondem à amostra como um todo. Porém, nem sempre os dados estão assim. Vamos abrir um banco de dados com pesos em quilogramas de crianças de uma creche, da turma A (20 crianças) e da turma B (3 crianças). É preciso perceber que apesar de termos valores de duas turmas, o banco foi criado com apenas duas variáveis: turma, a qual indicará a qual turma cada indivíduo pertence; peso, que é o valor visto na balança em quilogramas.

O banco está no formato *.tsv. Vamos abrir utilizando a função import() do pacote rio. Se você ainda não tiver esse pacote, instale-o antecipadamente.

```
#Abrindo o banco
install.packages('rio')
library(rio)
creche <- import('caminho_da_pasta/creche.tsv')
creche
   turma      peso
1      A  7.585529
2      A  7.709466
3      A  6.890697
4      A  6.546503
5      A  7.605887
6      A  5.182044
7      A  7.630099
8      A  6.723816
9      A  6.715840
10     A  6.080678 ...

#Cálculo da amplitude dos pesos na turma A
amplitude_A = max(creche[creche$turma == 'A', 'peso']) - min(creche[creche$turma ==
'A', 'peso'])

amplitude_A
[1] 3.635268

#Cálculo da amplitude dos pesos na turma A
amplitude_B = max(creche[creche$turma == 'B', 'peso']) - min(creche[creche$turma ==
'B', 'peso'])

amplitude_B
[1] 10.50057
```

Vamos entender o que escrevemos no código, passo a passo:

- amplitude_A: estamos criando um objeto para ficar salvo o valor da amplitude;

- max(creche[creche$turma == 'A', 'peso']): estamos querendo o valor máximo de um filtro que realizamos no banco chamado "creche";
- creche[creche$turma == 'A', 'peso']: estamos filtrando o banco "creche", selecionando apenas as linhas as quais a creche A é verdadeiro, e estamos selecionando a coluna chamada "peso", onde estão os valores quantitativos.

Pudemos ver que amplitude dos pesos na turma A foi menor do que a turma B (respectivamente, 3,64 kg e 10,50 kg). Isso indica que os dados da turma A variou menos do que a da turma B segundo as amplitudes.

Qual a melhor forma de calcular as amplitudes? Trabalhe as duas formas. Quem dirá serão os dados que você terá em mãos. Com o seu desenvolvimento na linguagem R, verá que há outras tantas formas de se trabalhar com isso.

7.2 INTERVALO INTERQUARTIL

O intervalo interquartil, ou amplitude interquartil (AIQ – visto no capítulo acima), ou desvio interquartílico é a diferença entre o 3° quartil e o 1° quartil (ver capítulo sobre quartis e *box-plot*).

$$AIQ = Q_3 - Q_1$$

Vamos construir um vetor de uma pequena amostra e ver uma das possibilidades para calcular o intervalo interquartil. O exemplo abaixo são dosagens de um exame chamado Proteína C Reativa (PCR), um clássico marcador inflamatório. Usaremos a função quantile() para sabermos os quartis desejados. Além disso, usaremos o esquema de colchetes para que o valor retornado pelo R seja puramente ele, "cru". Como curiosidade, teste logo depois sem os colchetes e veja o que irá retornar; o formato não será exatamente o mesmo.

```
pcr <- c(0.1,0.2,0.08,0.09,0.2,0.1,0.3,0.1,0.001,0.5,1.2,0.4,1.1,0.9)
quantile(pcr,0.75)[[1]] - quantile(pcr,0.25)[[1]]
[1] 0.375
```

Vamos exercitar em um banco de dados um pouco mais robusto? Nesse exemplo abaixo, banco chamado "dosagem_pcr.tsv", há dados de PCR de pacientes em processo inflamatório (casos) e pacientes normais (controles).

```
library(rio)
dosagem_pcr <- import('caminho_da_pasta/dosagem_pcr.tsv')
```

```
dosagem_pcr
       grupo           pcr
1   controle -0.0621197248
2   controle  0.3832287726
3   controle  0.6253323530
4   controle -0.4037093108
5   controle  0.4287374066
6   controle  0.4518167676
7   controle  0.1275780120 ...

AIQ_casos <- quantile(dosagem_pcr[dosagem_pcr$grupo == 'caso', 'pcr'],0.75) -
  quantile(dosagem_pcr[dosagem_pcr$grupo == 'caso', 'pcr'],0.25)

AIQ_casos
     75%
2.058794

AIQ_controles <- quantile(dosagem_pcr[dosagem_pcr$grupo == 'controle', 'pcr'],0.75) -
  quantile(dosagem_pcr[dosagem_pcr$grupo == 'controle', 'pcr'],0.25)

AIQ_controles
      75%
0.2846876
```

Perceba que houve um tal de "75%" na resposta que o R retorna. Os valores de AIQ estão perfeitamente corretos, porém, se você quiser jogar esse número de AIQ em uma outra função, pode haver erro. Por isso, nas situações semelhantes, quando o R retornar algo a mais do que o valor desejado, use colchetes duplos:

```
AIQ_casos[[1]]
[1] 2.058794

AIQ_controles[[1]]
[1] 0.2846876
```

Pelo cálculo acima, percebemos que o intervalo interquartil entre os casos é maior do que entre os controles, ou seja, os valores da Proteína C Reativa estão variando mais entre os casos. Para a gente enxergar melhor esse fenômeno, vamos fazer o *boxplot*.

```
#Figura 7.1
boxplot(peso ~ grupo, data = dosagem_pcr)
```

Quando a gente está falando de variação dos dados, no *boxplot* a gente dá ênfase ao tamanho das caixas. É nítido que caixa entre os casos é maior do que a caixa dos controles. Lembrando que dentro dessas caixas, há 50% dos dados para cada grupo. Assim, perceba que os dados entre os controles estão mais juntos, a caixa é pequena, então estão variando menos.

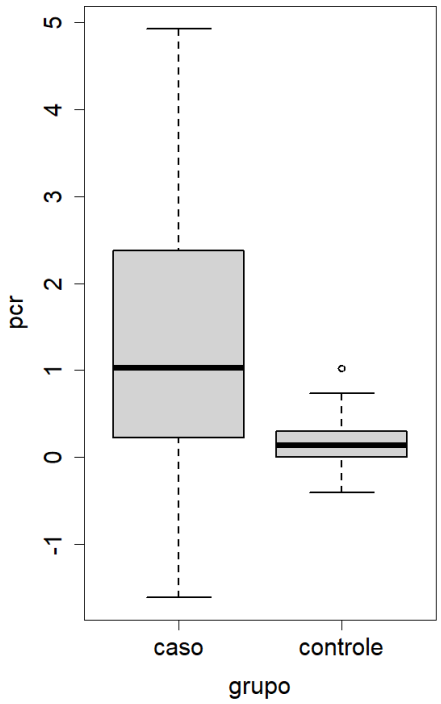


Figura 7.1. *Boxplot* de peso por grupo.

7.3 VARIÂNCIA E DESVIO PADRÃO

Antes de entendermos o que é desvio padrão e o porquê que os analistas comumente o utilizam, vamos compreender o que é o desvio absoluto e o desvio médio primeiro.

Imagine um certo conjunto de dados: peso de pacientes atendidos por um endocrinologista, por exemplo. Dificilmente os números vão se repetir, até porque normalmente tratamos esses dados com dois algarismos após a vírgula. Haverá pacientes abaixo do peso ideal, haverá pacientes com peso adequado e haverá pacientes com peso elevado. O que ocorre? Cada dado aqui levantado se afastará da média um tanto. Esse tanto é o que chamamos de desvio.

Dessa forma, **desvio absoluto** é a diferença entre cada dado e a média. Matematicamente teremos que:

$$desvio = xi - \overline{x}$$

onde *xi* representa cada observação. O sinal, se negativo ou positivo, aqui pouco importa. Vamos calcular no R o desvio dos dados fictícios abaixo:

```
peso1 <- c(89,47,62,48,102,74,52)   #em quilogramas
```

```
#criando função para calcular os desvios
desvio <- function(x){
  return(x - mean(x))
}

round(desvio(peso1),2)
[1] 21.29 -20.71 -5.71 -19.71 34.29 6.29 -15.71
```

Como interpretar esses resultados? Inicialmente, vamos calcular a média dos pesos:

```
mean(peso1)
[1] 67.71429
```

Então, a média de todos os pesos dos pacientes foi 67,71 kg. Pelos desvios gerados, o primeiro paciente desvia da média 21,29 kg. Já o segundo, tem um desvio de 20,71 kg. Foram desvios relativamente parecidos, só que o primeiro paciente tem um peso maior que a média, já o segundo paciente, tem um peso abaixo da média calculada. Vejamos o terceiro paciente, obteve um desvio de 5,71 kg. Esse foi o paciente que em peso mais se parece com a média, pois em módulo, obteve o menor desvio absoluto.

Quando a gente começa a se questionar se os desvios entre os pacientes são elevados, vamos tendo a ideia do quão importante são as medidas de dispersão, pois, se os dados variam muito, é bem possível que haja grandes desvios, correto? Por outro lado, se os dados dos pacientes fossem muito próximos, os desvios seriam bem pequenos e eles estariam bem pertos da média. Entenda: essa preocupação não é vã. Muitas análises estatísticas vão depender dessas variações e isso ajuda as suas análises a terem poder de teste.

Seguindo em frente, poderíamos nos questionar: por que não fazer uma média desses desvios? Perfeito! Até mesmo porque, intuitivamente falando, se essa média dos desvios estiver muito alta, haverá bastante variação entre os dados. Relembrando, para calcular a média, temos que somar os dados que desejamos e dividir pelo número de observações. Pois bem, começar pela soma dos desvios:

```
sum(desvio(peso1))
[1] -4.263256e-14
```

Não lhe causa estranheza esse número? Esse número está em notação científica no R e ele pode ser traduzido como 4,26 x 10^{-14} kg ou ainda 0,0000000000000426 kg. Na realidade, esse número é exatamente zero. Os algarismos 4263256, enfim, vêm de arredondamentos quando calculamos os desvios. Então, veja que é impraticável calcularmos a média dos devios, pois zero dividido por número maior que um, é zero. Dessa forma, o desvio médio é meramente sempre o valor zero. Não é informativo!

Então, precisamos calcular essa "média" utilizando os desvios sem os sinais. Os matemáticos resolveram elevar ao quadrado esses desvios. Dessa forma, eliminamos os valores negativos:

$$s^2 = \frac{\sum(x - \bar{x})^2}{(n-1)}$$

onde s^2 é a **variância**; x - $\bar{x}$ é o cálculo dos desvios. Aqui nós reduzimos n por uma unidade por conta dos graus de liberdade que precisamos ter para a variância. Então, de uma forma grosseira, a variância é como se fosse uma média dos desvios elevados ao quadrado, que é para não dar zero. Importante salientar que calculamos a variância amostral (dos dados que temos), mas estamos na verdade sempre intuindo a variância populacional (que só poderia ser calculada com todos os dados da população). Para diferenciar uma coisa da outra quando necessário, a variância populacional será indicada pela letra grega sigma ao quadrado: σ^2.

Sensacional! Resolvemos um problema, que é a média dos desvios darem zero. Mas geramos outro, por que a variância é dada ao quadrado, então, seu resultado é interpretado ao quadrado também. Vamos entender isso calculando a variância no R. Use a função var() para calcular a variância dos dados.

```
var(peso1)
[1] 460.9048
```

Dessa maneira, a variância foi 460,90 kg^2. O que é uma coisa com quilogramas ao quadrado exatamente? Uma loucura, não é? É difícil interpretar esse valor. Imagina que estamos trabalhando com dados de comprimento dos axônios de neurônios medulares (metros) e calculamos a variância: isso seria m^2, o que é unidade de área e estaria incorreta a sua interpretação.

Por conta disso, existe o **desvio padrão** (*s*), que é a raiz quadrada da variância.

$$s = \sqrt{s^2}$$

O desvio padrão populacional é indicado pela letra sigma somente (σ). Para calcular no R, podemos usar a função sqtr() (vem do inglês *square root*) que retorna a raiz quadrada, ou, diretamente usamos a função sd() (vem do inglês *standard deviation*) para o R retornar o desvio padrão dos dados.

```
#a raiz quadrada da variância
sqrt(var(peso1))
[1] 21.46869

sd(peso1)
[1] 21.46869
```

Perceba o óbvio, que esse número é menor do que a variância. Não esquecendo as unidades, o desvio padrão entre os pesos é 21,47 kg. Muito mais amigável, concorda? Dessa forma, os valores dos pesos dos pacientes se deslocam aproximadamente 21,47 kg da média.

Então, o desvio padrão, além de ser a raiz quadrada da variância, é uma medida de quanto os valores se afastam da média aritmética. Quanto maior o seu valor, maior a variação entre os dados existirá. Para comparamos dois conjuntos de dados e analisarmos qual dos dois há mais variação, vamos supor que em outro dia esse mesmo endocrinologista consiga o peso de outros sete pacientes, que chamaremos de peso2:

```
peso2 <- c(65,48,159,52,87,96,61)

sd(peso2)
[1] 38.58077
```

Portanto, os valores de peso da segunda coleta de dados variaram mais do que a primeira coleta.

É importante destacar aqui que essas comparações entre variações devem ser feitas com medidas exatamente iguais, por exemplo: estamos comparando variações de pesos. Não é adequado verificarmos se variou mais ou menos comparando dados de peso e dados de altura dos pacientes, pois os espectros dessas duas variáveis são diferentes (peso em quilogramas; altura em metros).

Quando você for expressar a média de uma determinada variável, coloque junto o valor do desvio padrão da seguinte forma: média ± desvio padrão. Exemplo: o peso médio dos pacientes foi relativamente alto (98,50 ± 3,54 Kg).

7.4 COEFICIENTE DE VARIAÇÃO

Você acompanhou logo acima que um dos problemas ao comparar o quão dados diferentes variam é que temos que comparar dados de mesmo espectro, de mesma unidade de medida. Mas, será que tem como compararmos dados de espectros diferentes? A resposta é sim. Utilizando um coeficiente que é dado em porcentagem. Dessa maneira, saímos das unidades específicas de cada variável e podemos verificar variações de dados de uma forma genérica.

O coeficiente de variação (CV) é o desvio padrão relativo à média. Como ele é relativo, podemos expressar seu valor em porcentagem. Vejamos:

$$CV = \frac{s}{\bar{x}} \cdot 100$$

O pacote básico do R não há nenhuma função nativa específica que calcule o coeficiente de variação, mas podemos calculá-lo facilmente escrevendo a fórmula acima em código. Vamos retomar o peso de paciente que visitaram hipoteticamente um endocrinologista, mas agora teremos uma variável a mais que são as suas alturas, em metros.

```
peso1 <- c(89,47,62,48,102,74,52)
altura <- c(1.83, 1.52, 1.54, 1.49, 1.87, 1.77, 1.63)

#coeficiente de variação dos pesos
cv_peso <- sd(peso1)/mean(peso1)*100
cv_peso
[1] 31.70482

#coeficiente de variação das alturas
cv_altura <- sd(altura)/mean(altura)*100
cv_altura
[1] 9.461419
```

Dessa forma, podemos verificar que os pesos variaram mais que as alturas (respectivamente, 31,70% e 9,46%). Se você reparar bem, a altura varia em casas decimais, enquanto que o peso varia em dezenas. Querendo ou não, a escala ainda é um complicador nesse sentido. Veremos em capítulos adiante como resolver essa questão e deixar os dados em escalas semelhantes e tirar conhecimentos mais precisos a partir disso na padronização dos dados.

Se você tiver que calcular vários coeficientes de variação, talvez fique inviável ter que escrever dezenas de centenas de linhas de comando. Talvez, criando uma função própria para isso resolva, da seguinte maneira:

```
cv <- function(x){
  coeficiente <- sd(x)/mean(x)*100
  return(coeficiente)
}

cv(peso)
[1] 31.70482

cv(altura)
[1] 9.461419
```

Uma curiosidade interessante está no hemograma, aquele exame de sangue, um dos mais requeridos na rotina laboratorial de Análises Clínicas. Dentro do hemograma há um parâmetro chamado RDW (do inglês, *Red cell Distribution Width*); aportuguesando, pode ser referido como Índice de Anisocitose Eritrocitária. Na realidade, esse parâmetro avalia a variação do volume das

hemácias. Quanto maior a variação do volume entre as células, maior será o RDW. Quando isso acontece, damos o nome de anisocitose, o que pode ser característico de algumas anemias.

Pois bem, esse RDW é justamente o coeficiente de variação calculado pela máquina que executa o exame. O equipamento tem o poder de medir o volume de cada hemácia que vai passando pelo seu detector. Com isso, ele calcula a média do volume dessas células, que é um outro parâmetro presente no hemograma, o VCM (Volume Corpuscular Médio). Se ela tem os dados dos volumes, por que não calcularia também o desvio padrão? Com a média e o desvio padrão, teremos facilmente o coeficiente de variação.

É, senhoras e senhores, há muita matemática na área da saúde.

Capítulo 8

Conceitos Básicos de Probabilidade

No Prefácio deste livro conto rapidamente a história do urubu cor-de-rosa. Caso você não tenha lido ou esqueceu, em poucas frases te mostro novamente. Um metre meu da universidade havia lançado uma pergunta para a classe: eu conseguiria ver um urubu rosa voando? Estamos propensos a responder que não, mas há várias maneiras em que um urubu rosa possa cortar os céus.

A questão aqui é a probabilidade de isso acontecer. Muito baixa, é verdade. Revalido que na área biológica praticamente tudo é uma questão de probabilidade. Portanto, relembrar e entender alguns pontos sobre o assunto é importe nesse nosso caminho, pois muitas análises e compreensões dos resultados dependem do básico sobre probabilidade.

Não há nada mais simples do que começar a explicar probabilidade usando uma simples moeda. Um velho, mas bom clichê, bem verdade. Se uma pessoa joga para cima uma moeda, a probabilidade de cair cara é 50%. A probabilidade de sair coroa também é 50%. Mas, de onde veio esses valores? Primeiro, precisamos saber o que é um evento, que é um fenômeno, um acontecimento, uma ocorrência. No caso acima, sair cara é um evento. Entretanto, existe um espaço amostral, que são todos os eventos possíveis, que no caso da moeda, pode ser cara ou cora (duas possibilidades). Dessa forma, a probabilidade para sair cara em uma moeda, temos um evento (sair cara) e universo amostral igual a dois (cara ou coroa).

Segundo a definição clássica, a probabilidade de se obter um evento é dada pela divisão desse evento pelo universo amostral, ou seja:

$$P(A) = \frac{E}{U}$$

onde P(A) indica a probabilidade de o evento acontecer, onde *E* é o número de eventos e *U* é o universo amostral. Sendo assim, a probabilidade de sair cara é calculada P(A) = 1/2, ou 0,5.

Vamos exercitar no R? Qual a probabilidade de dentro do alfabeto escolhermos aleatoriamente a letra K? Bom, de cabeça eu não faço a mínima ideia, mas podemos calcular facilmente. Vamos utilizar a função letters() do R para formar uma lista de todas as letras do alfabeto. Com a função length() nós contaremos quantas são (pelo menos isso eu sei de cabeça: são 26; mas o intuito é não depender muito da cabeça humana e conhecer novas funções da linguagem de computação). Sendo a letra K uma letra, temos um só evento, portanto:

```
1/length(letters)
[1] 0.03846154
```

Sendo assim, a probabilidade de escolhermos a letra K é de aproximadamente 3,85%. Perceba que para achar esse número em porcentagem bastou multiplicarmos por 100 do resultado gerado. Poderíamos inclusive escrever nosso código da seguinte maneira:

```
1/length(letters)*100
[1] 3.846154
```

O maior intuito aqui não é você sair calculando todas as probabilidades do mundo, mas entender melhor o que representa tais valores. Quando calculamos a probabilidade a partir da fórmula, o valor de saída é um número entre 0 e 1; no R dificilmente não será assim. Essa amplitude será muito utilizada na jornada estatística, porém, quando formos falar sobre nossos resultados (em um trabalho científico, por exemplo), é mais conveniente mostrar ao leitor do seu texto esse valor em porcentagem, pois nós humanos temos melhor compreensão da probabilidade dessa maneira (ou, nos acostumamos assim).

Assim sendo, a probabilidade de 0 é 0% de acontecer (então, é certeza que não ocorrerá). Já a probabilidade de 1 é mesmo que 100% (então, certamente acontecerá). No caso da moeda, ocorrer cara em um lançamento é 0,5 ou 50% de chance de acontecer (então, pode acontecer ou não; a chance de ocorrer ou não é exatamente a mesma).

8.1 ACRÉSCIMO/DECRÉSCIMO PERCENTUAL SOBRE UM VALOR

Um fato muito comum na área financeira, por exemplo, é o acréscimo ou decréscimo de X% em cima de algum valor monetário. Exemplo: as ações de uma empresa subiram 15% em relação ao mês anterior; a taxa de juros teve um decréscimo de 1,75% em relação ao mesmo período do ano passado; e por aí vai. Apesar de não ser tão frequente assim na área biológica quanto na área financeira, podemos usar esse conhecimento a nosso favor e por vezes haverá situações em que expor essas taxas percentuais será fundamental para o entendimento, principalmente, epidemiológico. Exemplo: houve elevação de 34% na frequência de internação hospitalar por miocardite em relação ao ano passado (é preciso averiguar o porquê disso, concorda?).

Uma dica fundamental é perceber que nesses casos o tempo/período é uma coisa importante. Nos exemplos acima, perceba que foi estabelecido “em relação ao mês passado”, “em relação ao ano anterior”. Se quisermos saber diferenças percentuais, temos que ter pelo menos dois números que correspondam a tempos/períodos diferentes (valor do início e valor no fim).

Vamos supor que um produtor de porcos esteja preocupado na engorda de seus animais. Ele pesou um leitão e na pesagem deu 4,35 kg. Na semana seguinte, esse leitão foi novamente

pesado e deu 5,63 kg. O óbvio é que ele engordou, porém, ele engordou quantos por cento? Ter esse conhecimento em mãos é interessante para sabermos se a ração está adequada ao que se espera de engorda; se esse percentual de ganho condiz com a raça ou genética do animal; se não for o esperado, tomar estratégias para intervenção (mudança da alimentação, por exemplo, ou outros fatores).

Para isso, podemos usar uma fórmula, que é simples e podemos guardar de cabeça para usarmos em diversas situações do cotidiano, como compras no supermercado:

$$taxa = \frac{último\ valor - primeiro\ valor}{primeiro\ valor}$$

Sabendo disso, vamos para o R calcular a taxa de engorda do referido leitãozinho:

```
taxa_1 <- (5.63 - 4.35) / 4.35
taxa_1
[1] 0.2942529

#transformando em %
taxa_1 * 100
[1] 29.42529
```

Olhando os resultados, o leitão engordou a uma taxa de 0,2943, ou 29,43% em uma semana.

Vamos para outro exemplo. Segundo estimativas do Censo brasileiro, no ano de 2000 éramos 169.590.693 habitantes. Dez anos depois, fomos para 190.755.799 habitantes. Com certeza houve um acréscimo, mas de quantos por cento? Fecha o livro e tenta fazer essa conta antes de seguirmos.

Espero que você tenha acertado. Vamos ver no R como seria:

```
#populacao brasileira
taxa_BR <- (190755799 - 169590693)/169590693
taxa_BR * 100
[1] 12.48011
```

Pois bem, entre 2000 e 2010, houve um acréscimo de 12,5% na população brasileira. Quando a taxa retornada é um valor negativo, significa que houve decréscimo. Por exemplo, uma pessoa está se exercitando e inicialmente ela tinha 89,5kg, e depois de seis meses apresentava 75,2kg. Ela reduziu 14.3 kg, mas isso é uma redução percentual de 15.98%. Perceba:

```
taxa_peso <- (75.2 - 89.5) / 89.5
taxa_peso
[1] -0.1597765
```

```
#transformando em %
taxa_peso * 100
[1] -15.97765
```

Com isso exposto, iremos agora para um outro conhecimento: o fator multiplicativo. Diversas vezes nos deparamos com promoções de 10%, 15% ou 20%. Você já sabe achar o valor que irá pagar com base no valor original do produto? Iremos exemplificar com uma situação corriqueira do nosso dia a dia. Vamos supor que queremos um produto que custa 25,31 reais, e a loja afirma que para esse produto há um desconto de 20%. Desta forma, retiraremos 20% do valor original.

Existe diversas formas para fazer isso, porém, vou aqui demonstrar a forma que considero mais simples e que você pode aplicar na sua rotina. Sabendo o fator de multiplicativo (que é simples, você verá), evitamos fazer muitas contas. Assim, pegamos o valor original, multiplicamos por esse fator multiplicativo e o valor desejado aparecerá. Mas como achar esse fator?

Existe uma sutil diferença entre o fator multiplicativo de acréscimo para o de decréscimo, mas é intuitivamente simples que percebermos que ao acrescentar um percentual o valor original aumenta, enquanto que retirar um percentual, o valor original irá diminuir. Pois bem, para acharmos o fator multiplicativo de acréscimo de 30%, fazemos:

Taxa de acréscimo de 30% = 30/100 = 0,30

Fator multiplicativo de acréscimo de 30% = 1 + taxa de acréscimo = 1 + 0,30 = 1,30

Então, o acréscimo de 30% de 500 kg de peso de um animal, basta multiplicar:

500 x 1,3 = 650 kg

Se uma pessoa de 75 kg engordou 20% do seu peso em um determinado período, então, atualmente ela está com 90 kg, pois:

75 x 1,2 = 90 kg

Se em um município choveu nesse mês 80% a mais que a média mensal normal que é de 150 mm de chuva, então nesse específico mês choveu 270 mm de chuva.

150 x 1,8 = 270 mm

Já o fator multiplicativo de decréscimo é calculado reduzindo a taxa que se estipula. Exemplo: um animal doente teve decréscimo de 30% do seu peso inicial (que era 430 kg). Dessa forma,

Taxa de decréscimo de 30% = 30/100 = 0,30

Fator multiplicativo de decréscimo de 30% = 1 – taxa de decréscimo = 1 – 0,3 = 0,7

430 x 0,7 = 301 kg

Se um instrumental médico custa R$ 12.300,00, mas o vendedor está dando desconto de 10%, o valor de compra é de R$ 11.070,00, pois:

12300 x 0,9 = 11070

Se uma população de 3 milhões de habitantes houver um decréscimo de 1,3% por conta de óbitos por um vírus letal, no segundo período essa população terá 2.961.000, uma vez que:

3000000 * (1-0.013) = 3000000 * 0.987 = 2961000

Particularmente, eu gosto de fazer essas contas rapidamente nos supermercados, quando eles mostram descontos fáceis de calcular, como 10%, 15% e 20%. Respectivamente, os fatores multiplicativos são 0,9, 0,85 e 0,8. Multiplicando pelo valor do produto, eu sei quanto irei pagar no caixa (e nem sempre a promoção é a melhor opção de compra, pois pode haver outras marcas com valor mais barato do que o produto promocional).

8.2 PROBABILIDADE BASEADA NA FREQUÊNCIA RELATIVA

Ainda não abordamos essa questão sobre frequências, que epidemiologicamente falando, é essencial na avaliação de dados categóricos. Porém, uma breve introdução será de fácil entendimento para que entendamos a relação que pode ser feita entre frequências relativas e a probabilidade de um determinado evento acontecer.

Pois bem. Em um estudo foram selecionados dois mil indivíduos de um determinado município, dos quais 20 apresentavam lúpus eritematoso (uma doença autoimune) e 1.980 não apresentavam qualquer sintoma relacionado. Tais números, 20 e 1.980 são as frequências chamadas absolutas, porém, podemos representar esses números em porcentagem, transformando-os em frequências relativas.

Para isso, dividimos o valor absoluto de uma classe pelo total de indivíduos. Dessa forma:

```
#individuos com lúpus eritematoso
20/2000
[1] 0.01

#indivíduos sem a condição
1980/2000
[1] 0.99
```

Do jeito que escrevemos o comando, temos as frações numéricas respectivas, que tenho certeza que você já consegue interpretar o que está ocorrendo aqui. Contudo, como foi dito anteriormente, transformar esses valores em porcentagem ajuda o leitor a compreender melhor esses resultados. Portanto, vamos multiplicar cada um por cem:

```
#individuos com lúpus eritematoso
20/2000 * 100
[1] 1

#indivíduos sem a condição
1980/2000 * 100
[1] 99
```

Pronto! Temos os valores percentuais de cada classe, ou seja, 1% dos 2 mil indivíduos apresentavam lúpus, enquanto que os outros 99% não demonstravam essa condição.

Aqui começamos a captar o sentido que isso tem a nível probabilístico, pois, se essa amostragem fosse representativa da população desse município, poderíamos nos perguntar: qual a probabilidade de uma pessoa ter lúpus eritematoso em tal município? A probabilidade é de 1%. Assim como a probabilidade de não ter é de 99%.

A pergunta pode ser sensivelmente diferente: se, ao acaso, sortearmos um morador desse determinado município, qual a probabilidade de ele ter lúpus eritematoso? A resposta você já sabe, que é o mesmo 1%, ou, 1 a cada 100 moradores.

Quando falamos em Epidemiologia, Vigilância Epidemiológica, Saúde Coletiva, enfim, veremos muitos dados tratados desta forma.

8.3 OS TESTES DE HIPÓTESE E SUAS PROBABILIDADES

Um teste de hipótese é uma técnica estatística usada para tomar decisões sobre uma população com base nas informações de uma amostra. É o que corriqueiramente fazemos na área biológica, pois coletar dados de todos os indivíduos pode ser uma tarefa extremamente difícil e custosa financeiramente. Por isso, daí vem o termo inferência estatística, uma vez que estamos tomando como verdade o que ocorre em uma população os resultados vindos da observação de alguns indivíduos dela, não todos.

Portanto, na realidade, nos testes estatísticos estamos querendo dizer: com base na amostra que tenho, estou inferindo/deduzindo que o que ocorre com esses indivíduos da amostra, ocorre de igual forma a todos os outros indivíduos da população. É sempre uma tentativa de

generalização, mas nessa frase, não há como ter certeza dessa verdade. Há intrinsecamente em cada teste estatístico a probabilidade de estamos certos e a probabilidade complementar de estamos errados (mesmo que pequena quando tudo bem planejado, é inevitável).

Todo teste estatístico envolve a formulação de duas hipóteses opostas:

- a hipótese nula ou H_0 (lê-se: h zero), que representa uma posição padrão ou ausência de efeito, mas principalmente de igualdade;
- hipótese alternativa ou H_1 ou H_a, que representa a oposição à hipótese nula; ou, é fora do padrão, ou existe efeito, ou principalmente há desigualdade.

Um exemplo simples. Em uma pesquisa, uma veterinária deseja saber se suínos do sexo masculino têm a mesma massa óssea do que os indivíduos do sexo feminino. Vamos às hipóteses:

H_0: a massa óssea de machos é igual à massa óssea de fêmeas;
H_1: as massas ósseas são diferentes entre os sexos.

Não importa nesse momento o teste estatístico a ser realizado. Independente de qual seja, eu preciso aceitar uma hipótese e rejeitar a outra obrigatoriamente. Não há a chance de ter verdade estatística na comunhão das duas, pois são mutuamente excludentes. Dessa forma, um teste de hipótese avalia se os dados da amostra fornecem evidências suficientes para rejeitar a hipótese nula em favor da hipótese alternativa ou não. Portanto, estatisticamente, estamos na verdade mais preocupados com H_0, aceitando ou rejeitando-o.

Isso é feito considerando o nível de significância estatística, que representa a probabilidade de rejeitar a hipótese nula quando ela é verdadeira, ou seja, um tipo de erro. Na bioestatística, podemos escolher qualquer nível de significância, entretanto, 5% se tornou um padrão. Entenda o que esse número quer informar: existe a possibilidade de eu rejeitar H_0 quando essa hipótese é verdadeira, mas eu só tenho a probabilidade de 5% disso ocorrer. Conto com os outros 95% de probabilidade para que a decisão que eu tome esteja correta, rejeitando H_0 quando realmente devo rejeitá-lo. É por conta disso que a estatística é fundamentalmente probabilística.

Existem duas maneiras para avaliarmos se a hipótese nula deve ser aceita ou não. Há o método do valor crítico e o método do valor P. Nesse subcapítulo, falaremos basicamente sobre o método do valor crítico, para que entendamos como alguns testes são considerados. Entretanto, o método do valor P é o mais utilizado atualmente e será corriqueiramente abordado ao longo do livro (e conversaremos sobre ele mais à frente).

Sobre o método do valor crítico, muitos alunos ao começar os estudos na bioestatística acham um terror estudar isso. Mas, na realidade, é um passo a passo simples de se fazer. Cada teste estatístico precisa ser calculado um valor de teste. Se esse valor calculado for maior do que o limite pré-estabelecido pela tabela de distribuição de probabilidades (exemplificarei a seguir), então o teste é considerado significativo, ou seja, rejeitamos H_0. Caso contrário, aceitaremos H_0, o teste não é significativo.

Como falei, cada teste terá o seu valor calculado e associado a isso uma distribuição de números probabilisticamente. Existe a distribuição Z (normal), a distribuição t de Student, a distribuição qui-quadrado, a F de Fisher-Snedecor, binomial, de Poisson, e outras mais. A distribuição Z, por exemplo, tem a conformação de um sino em um gráfico (veja Figura 8.1) e todos os valores que a contém são possibilidades de algum evento ocorrer.

Na Figura 8.1 temos hachurada a região de rejeição de H_0, segundo o nível de significância escolhido. Nesse exemplo, estamos com duas regiões hachuradas, nas extremidades. Isso é típico de teste bicaudal, o que usamos em grande parte dos testes bioestatísticos. Se o valor de teste cair fora da região hachurada, aceitamos a hipótese nula (teste não significativo). Caso entre em alguma região hachurada, devemos rejeitar a hipótese nula (teste significativo).

O método do valor crítico depende também dos graus de liberdade. Na estatística, os graus de liberdade indicam o número de observações independentes que podem variar em um conjunto de dados de um experimento. Eles representam a quantidade de informações independentes disponíveis para estimar ou testar a hipótese nula. Em termos gerais, quanto maior o número de graus de liberdade, melhor a precisão e confiabilidade das análises estatísticas. Isso ocorre porque o número de graus de liberdade está diretamente relacionado ao n amostral. Outra peculiaridade é que cada teste tem o seu cálculo de graus de liberdade, que pode depender do número de grupos em análise também.

Vamos exemplificar o método do valor crítico realizando um teste simples de comparação de médias, chamado de teste t. Nele, testaremos se o comprimento da sépala da planta íris são iguais entre duas espécies: a *Iris virginica* e *Iris versicolor*. Vamos escrever as hipóteses primeiramente:

H_0: as médias dos comprimentos de sépalas entre as espécies são estatisticamente iguais.

H_1: as médias dos comprimentos são estatisticamente diferentes.

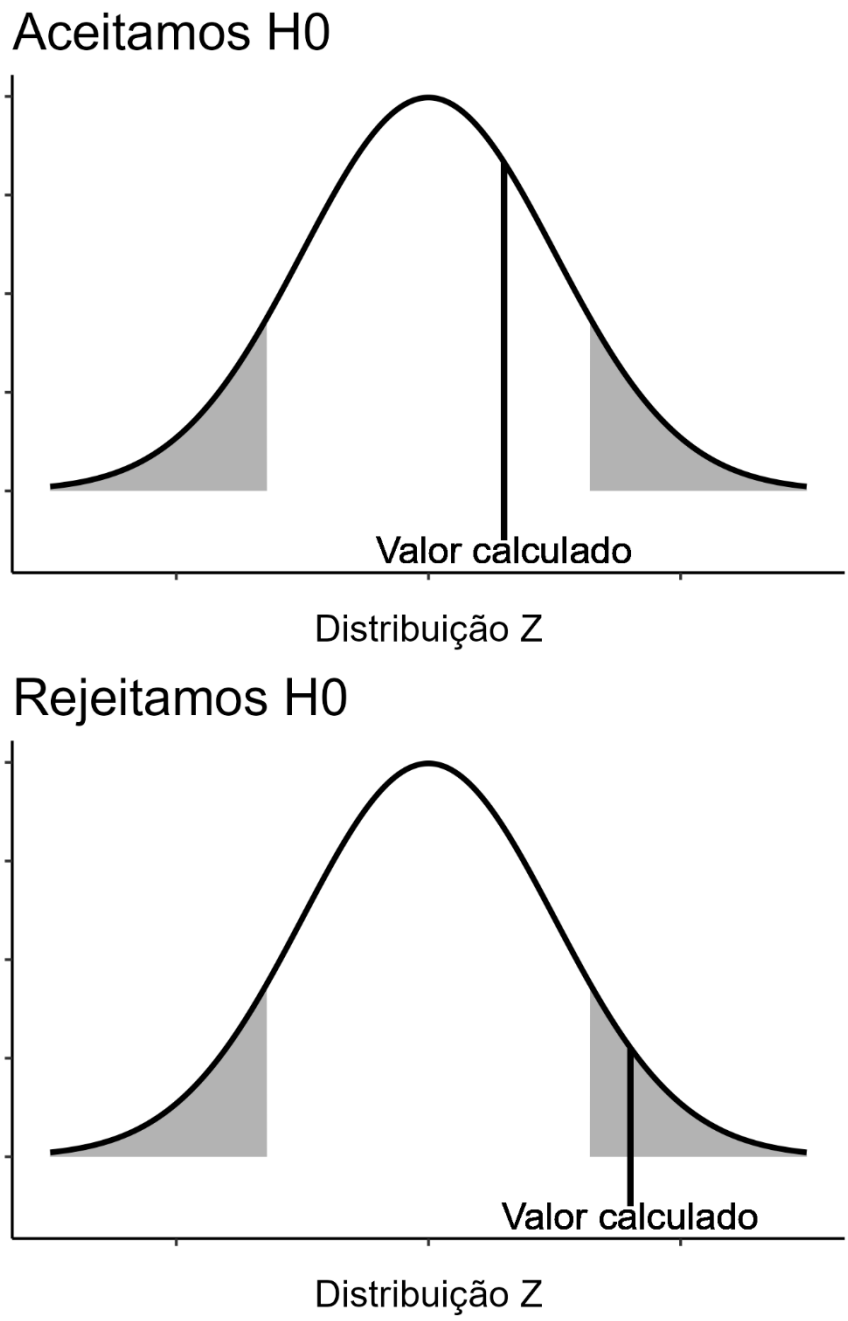


Figura 8.1. Distribuição Z, suas zonas de valores críticos e conclusão sobre H_0.

Para testar H_0, primeiro abriremos o banco com a função data(), pois esse banco de dados é nativo do R. Vamos filtrar os dados para cada espécie. Calcularemos o teste t usando a função t.test() e salvaremos em um objeto para verificarmos os resultados logo após, que chamaremos de teste_t.

```
#abrindo banco
data("iris")

#Filtrando dados das espécies virginica e versicolor
```

```
virginica <- iris$Sepal.Length[iris$Species == "virginica"]
versicolor <- iris$Sepal.Length[iris$Species == "versicolor"]

# Calcular o teste t
teste_t <- t.test(virginica, versicolor)

#Resultados
teste_t
      Welch Two Sample t-test

data:  virginica and versicolor
t = 5.6292, df = 94.025, p-value = 1.866e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.4220269 0.8819731
sample estimates:
mean of x mean of y
    6.588     5.936
```

A primeira coisa que temos como resultado é o valor de t calculado, que foi 5,6292. Os graus de liberdade são 94,025 (em df, do inglês *degrees of freedom*). O que falta avaliar são os limites tabelados da distribuição t para vermos se o valor calculado t vai além desses limites ou não. Para achar os valores tabelados, podemos utilizar a função qt() do R e indicar nela o nível de significância e os graus de liberdade que estamos trabalhando. Já temos essas informações, só que como o teste é bicaudal, o nível de significância de 5% será 2,5% para um lado e 2,5% para o outro (que deixa as nossas análises até mais conservadoras). Por conta dessa divisão, usaremos na função os valores de 0,025 e 0,975 (complementar do anterior). O que esperamos é encontrar valores matematicamente opostos em sinal. Veja:

```
#graus de liberdade
teste_t$parameter
      df
94.02549

#valores tabelados (bicaudal)
qt(0.025, df = teste_t$parameter)
[1] -1.985516

qt(0.975, df = teste_t$parameter)
[1] 1.985516

#valor de t calculado
teste_t$statistic
       t
5.629165
```

Então, por fim, achamos o limite de 1,985516. Vamos pôr todas essas informações em um gráfico e tirar a nossa conclusão final através da Figura 8.2.

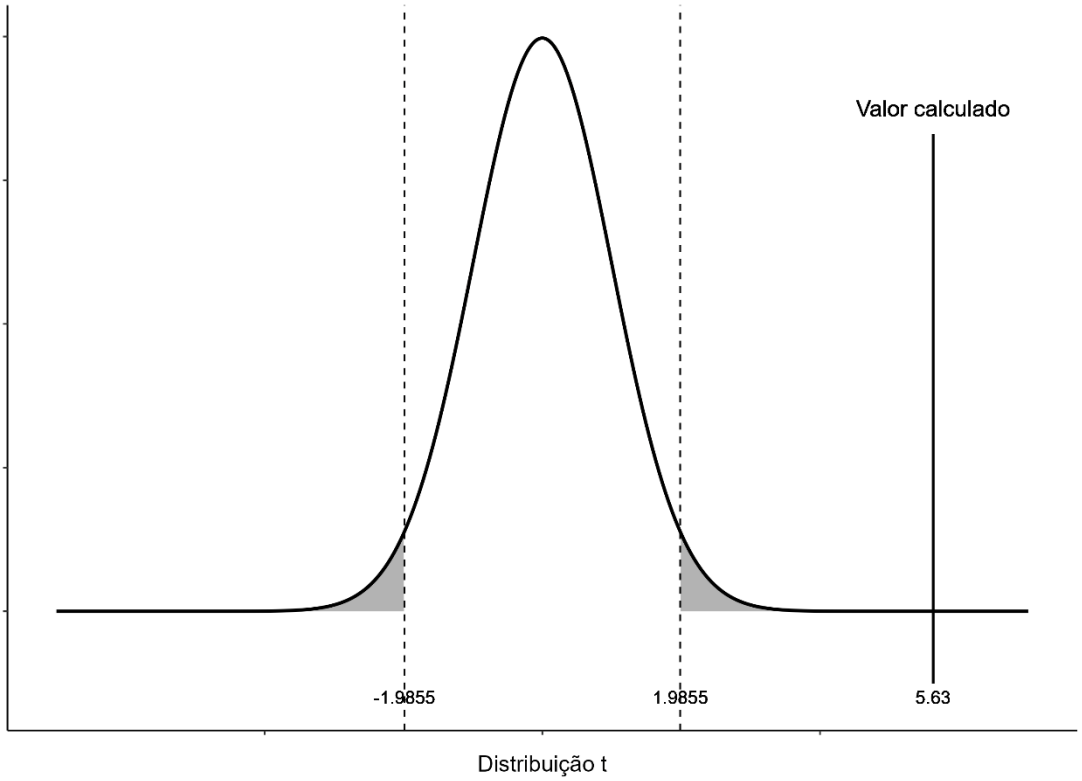


Figura 8.2. Esquema com os valores tabelados e o calculado do teste t.

Concluindo, como o valor calculado abrange a área hachurada, uma vez que passa do limite do valor tabelado de 1,9855, rejeitamos H_0 e podemos dizer que as médias de comprimento das sépalas são estatisticamente diferentes entre as duas espécies.

Por fim, quero lhes mostrar como o nível de significância pode influenciar na decisão de rejeição ou aceite do H_0. Observando a Figura 8.3, perceba que quanto maior o nível de significância, maior é a área de rejeição. Então, ao nível de significância de 10% fica mais fácil de observarmos diferença estatística, concorda? Enquanto que ao nível de significância de 1%, não seria possível rejeitarmos H_0.

O nível de significância é também chamado de α (alfa) que, como comentamos acima, é um tipo de erro probabilístico. Então, quanto maior o α, mais me exponho ao risco de rejeitar H_0 quando não deveria.

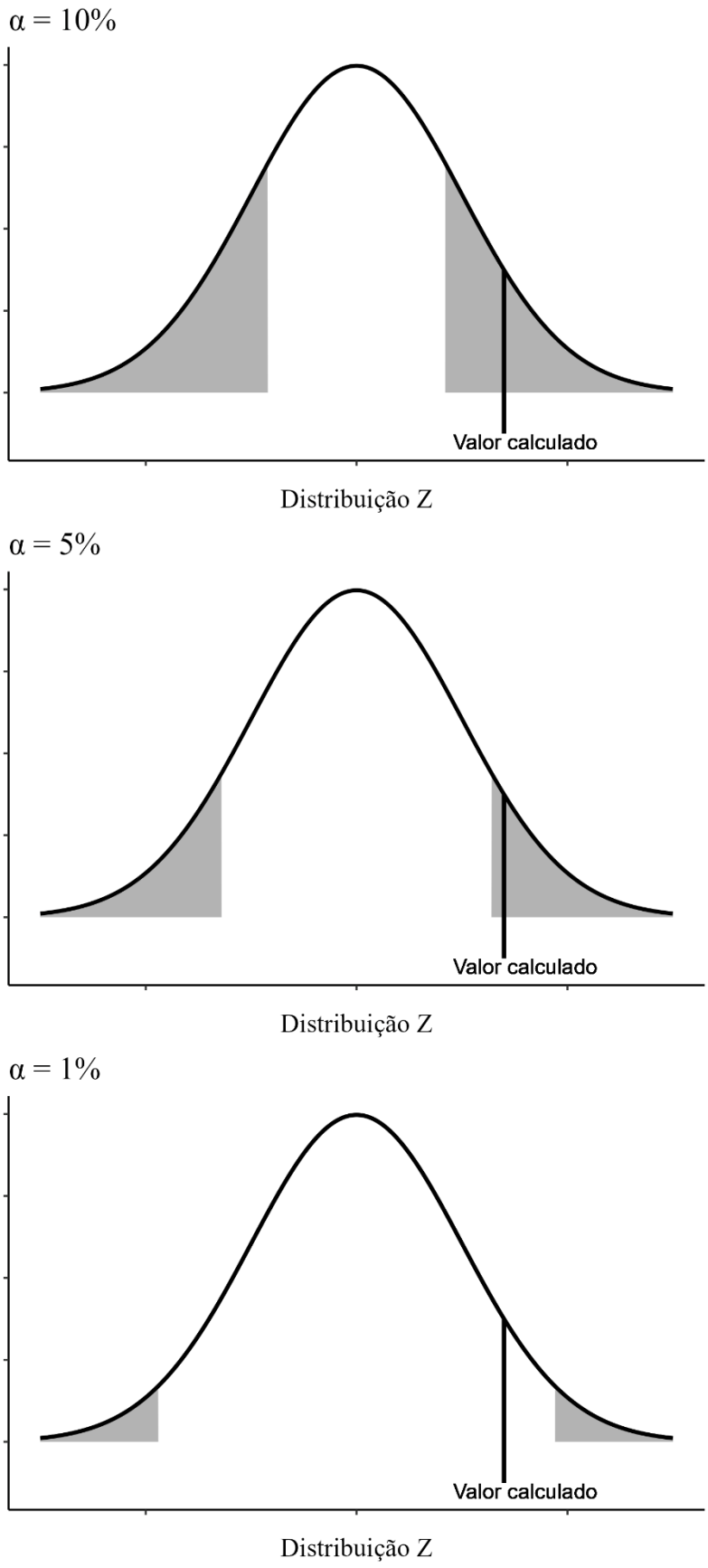


Figura 8.3. Diferentes níveis de significância e suas áreas de rejeição de H_0.
Nota: o valor calculado é exatamente o mesmo nas três situações.

8.4 O TAL DO P-VALOR

O outro método para avaliar a significância de um teste é pelo método do valor P, ou p-valor, ou P, ou ainda *p-value*. O primeiro passo é sempre a formulação das hipóteses. Por exemplo, acredito que a média de altura das crianças do terceiro ano da escola X é igual a média de altura das crianças do terceiro ano da escola Y, que fica em outro bairro. Então temos:

H_0: as médias das alturas dos alunos do terceiro ano são iguais.
H_1: as médias das alturas dos alunos do terceiro ano são diferentes.

Lembre-se que quando realizamos os testes de hipótese, estamos avaliando sempre H_0. Ou a gente aceita H_0, ou a gente rejeita H_0. Vamos supor que realizamos o teste t de *Studant* e o resultado dele nos indique que devemos aceitar H_0, ou seja, de que as médias das alturas são iguais estatisticamente. É importante que você saiba que há uma probabilidade embutida de estarmos certos, mas há também uma probabilidade embutida de estarmos errados. O p-valor se baseia em uma probabilidade e como se trata de uma probabilidade, o p-valor sempre vai variar entre 0 e 1.

Portanto, o p-valor é uma medida estatística usada para avaliar a evidência contra uma hipótese nula. Ele representa a probabilidade de se obter um resultado tão extremo quanto o que foi observado, assumindo que a hipótese nula seja verdadeira. Ou seja, um p-valor pequeno sugere que os dados observados são menos prováveis de ocorrer sob a hipótese nula, o que significa que há evidências mais fortes contra a hipótese nula. Quanto maior o seu valor, mais evidência a favor da hipótese nula.

Então, a decisão de rejeitar ou não a hipótese nula (H_0) com base no p-valor depende do nível de significância escolhido previamente. Se escolhermos o nível de significância de 5%, o nosso limite do P será 0,05. Se escolhermos o nível de significância de 10%, o limite seria de 0,1 e caso fosse estabelecido o nível de significância de 1%, o limite seria 0,01.

Se o p-valor for menor ou igual que o nível de significância ($P \leq \alpha$), então temos evidência estatística suficiente para rejeitar a hipótese nula em favor da hipótese alternativa. Isso significa que a diferença observada entre as amostras é improvável devido ao acaso, e é provável que haja uma diferença real entre as populações. P-valor de 0,001523 é um exemplo onde devemos rejeitar H_0 ao nível de significância de 5% ($P \leq 0,05$).

Por outro lado, se o p-valor for maior que nível de significância ($P > \alpha$), não temos evidência estatística suficiente para rejeitar a hipótese nula. Isso significa que a diferença observada entre as amostras pode ser devido ao acaso, e não há evidência suficiente para afirmar que há uma

diferença real entre as populações. P-valor de 0,7832 nós aceitaríamos H_0, ao nível de significância de 5% (P > 0,05).

De forma bem resumida:

- Se o p-valor ≤ 0,05, rejeitar H_0 → o teste é significativo.
- Se o p-valor > 0,05, aceitar H_0 → o teste não é significativo.

Quanto menor é o p-valor, mais significativo ele é. Entretanto, quando houver valores próximos ao valor limítrofe, não dê tanta importância assim ao ponto de corte, pois trata-se de um número arbitrário. Um p-valor = 0,055 ou p-valor = 0,045 são conclusões muito parecidas e não opostas.

Como conselho, caso um teste se aproxime muito do α (0,05) mas a conclusão seja de aceite de H_0, comente que é sugestiva a rejeição se ela fizer sentido no contexto biológico. Seja sincero(a) com o seu público e discuta o porquê de ser sugestivo.

Por exemplo, no teste de hipótese que montamos acima, se o p-valor resultante fosse 0,07, poderíamos dizer que apesar do p-valor não ter sido significativo, as médias das alturas dos alunos do terceiro ano são sugestivamente diferentes por conta da diferença étnica estar evidente, suponhamos.

Capítulo 5

A Distribuição Normal

Há na natureza um fenômeno recorrente na distribuição de dados, praticamente um padrão que ocorre com os valores de inúmeras variáveis numéricas. Já falamos sobre esse fenômeno em capítulos anteriores, porém, aqui vamos destrinchar melhor tudo isso para entendermos mais à frente como a estatística usa isso a nosso favor. Para ser prático, vamos gerar no R quantidades de dados aleatórios diferentes e perceber o que está acontecendo.

Para isso, usaremos a função rnorm(), na qual basta atribuir dentro dela um número que será a quantidade de números que queremos gerar:

```
rnorm(5)
[1]  0.5171631  0.2368634 -0.7068772  0.8189387  0.1100311
```

Com isso testado, vamos colocar esses números em um gráfico chamado histograma usando a função hist(), que já utilizamos anteriormente também. Entretanto, vamos gerar quantidade de números diferentes: 5, 10, 50 e 100. E observamos como os valores se distribuem no histograma.

```
#Figura 9.1
par(mfrow=c(2,2))    #função que junta gráficos
hist(rnorm(5))
hist(rnorm(10))
hist(rnorm(50))
hist(rnorm(100))
```

A função par() juntará os quatro gráficos que iremos construir, dando origem à Figura 9.1 logo abaixo. Nela, podemos perceber que quando geramos 5 números aleatórios o gráfico ficou bem disperso (Figura 9.1-A). Aumentando para 10 números, o gráfico ficou mais preenchido (Figura 9.1-B). Quando geramos 50 números aleatórios, a maioria deles ficou entre 0 e 1, gerando uma coluna alta, enquanto que nos extremos do gráfico as colunas estão baixas (Figura9.1-C). Quando geramos 100 números, o formato se repetiu: coluna central mais alta e colunas extremas mais baixas (indicando poucos números extremos – Figura9.1-D).

Se você rodar o comando acima, talvez haja algumas diferenças nos gráficos, porque estamos gerando números aleatórios diferentes. Mas se você for além, aumentando os valores em rnorm(), a tendência será similar. Em uma grande quantidade de situações, os dados obtidos de pesquisas gerarão esse comportamento sinuoso, deduzido na Figura 9.2. Essa é a chamada distribuição normal, a qual tem um formato de sino e é simétrica em relação ao eixo vertical que passa pela sua média.

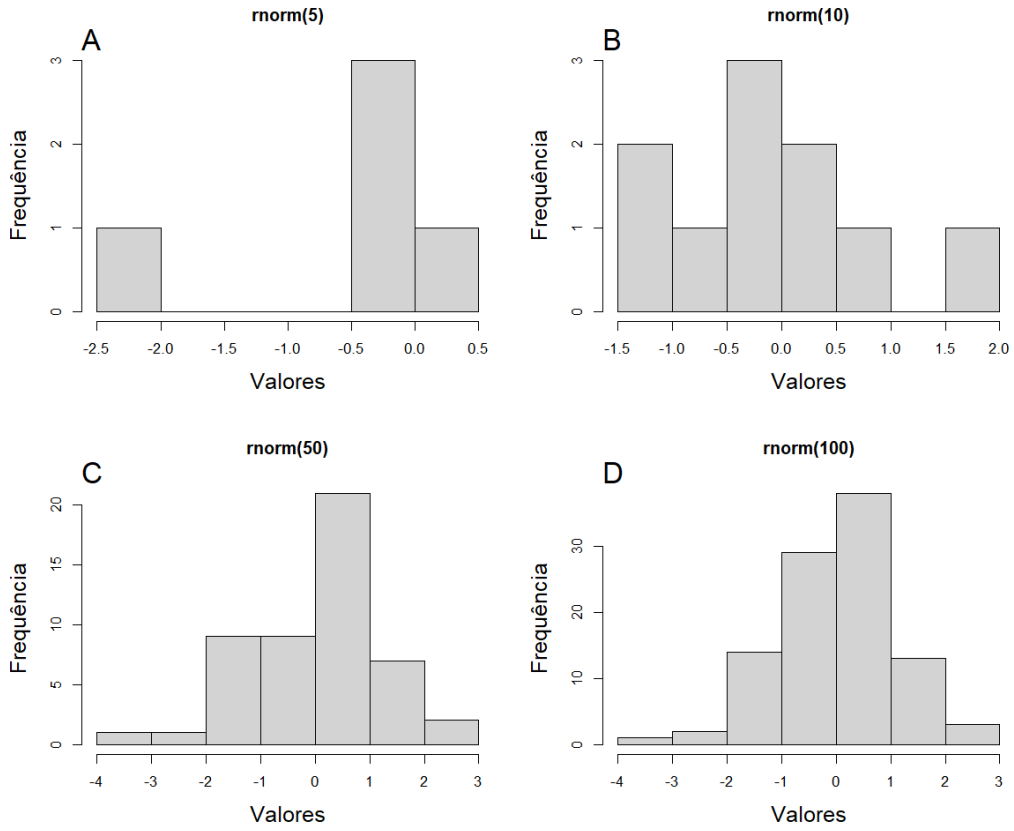


Figura 9.1. Histogramas representado distribuições quantitativamente diferentes.

Essa tendência vista graficamente pode ser chamada de curva normal ou curva gaussiana, nomeada assim em homenagem ao matemático Carl Friedrich Gauss, que foi um dos primeiros a descrever e estudar a distribuição normal.

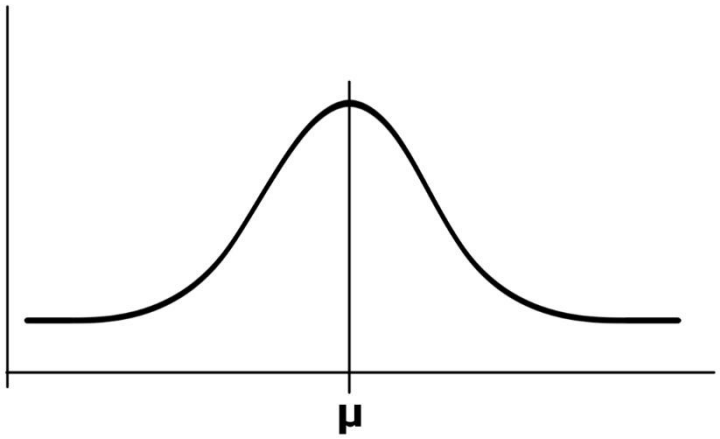


Figura 9.2. Distribuição normal.

Os estatísticos utilizam esse comportamento para realizar inúmeras análises. Só que para isso, precisou-se padronizar uma distribuição normal para que ela se encaixasse nos dados do dia a dia. Com isso, surgiu a **distribuição normal-padrão**, a qual tem algumas peculiaridades:

- O gráfico em forma de sino e simétrico;
- Média igual a zero ($\mu = 0$);
- Desvio padrão igual a um ($\sigma = 1$)

Na prática, isso não nos importa muito no momento, mas sim saber se os nossos dados se encaixam nesse tipo de distribuição ou não. Isso sim tem grande importância prática, uma vez que a análise correta será indicada dependendo da distribuição dos dados das variáveis em jogo.

Mas antes de discutimos isso melhor, você precisa saber que existe outros tipos de distribuição, ou seja, outros tipos de comportamento os quais os dados costumam apresentar. A normal é apenas uma delas, a mais frequência sim, mas não a única. Há a distribuição de Poisson, Binomial, Exponencial, de Weibull, entre outras. Porém, não faz parte do escopo deste livro detalhar nenhuma delas, mas sim sabermos o que corriqueiramente precisamos fazer e reconhecer quando a distribuição não é a normal. Primeiro indício é quando o gráfico não apresenta a característica de sino. Portanto, vamos recorrer à Figura 9.3, a qual exprime a simetria dos dados, seja na normal ou não.

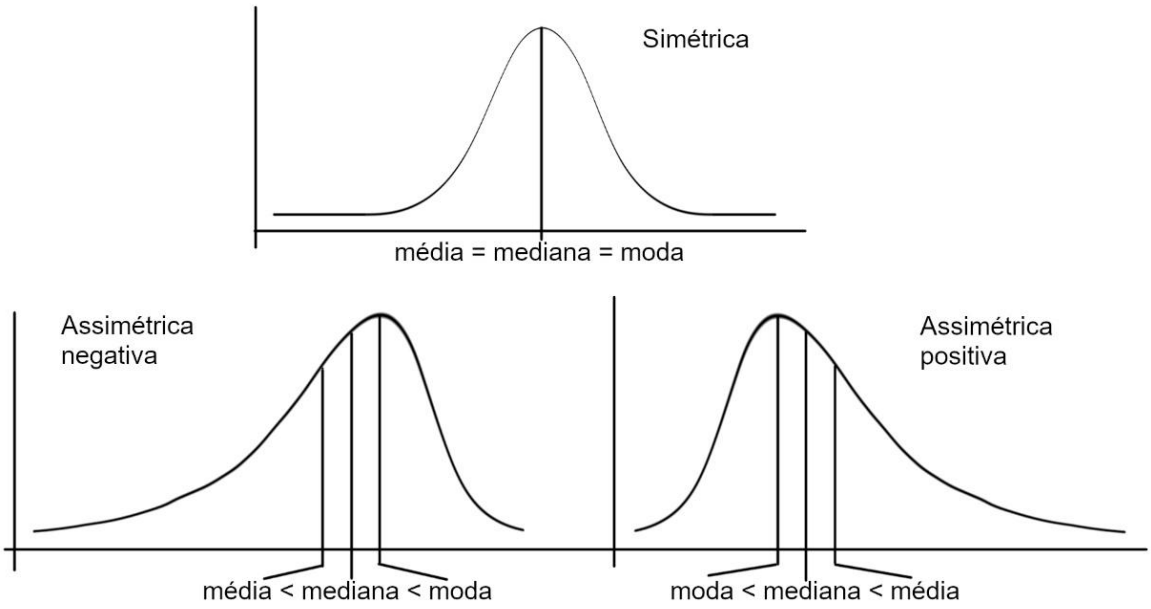


Figura 9.3. Simetria de distribuições de dados.

Primeiramente, quando a distribuição é normal, há simetria e a média, mediana e a moda costumem ficar bem próximas (parte superior da Figura 9.3). Já as distribuições que estão fora da normal, na área biológica os dados costumam perder a sinuosidade puxando uma "perna" do gráfico para um dos lados (parte inferior da Figura 9.3). Existem aqui duas situações:

1. Assimetria negativa, quando há muitos dados de pequeno valor em relação à moda. A extremidade esquerda do gráfico tem uma declividade mais suave. Nessa configuração, a média é altamente influenciada por esses valores mais baixos e acaba tendo valor menor do que a mediana e a moda.
2. Assimetria positiva, quando há muitos dados maiores do que a moda. A extremidade direita do gráfico tem declividade suava. Aqui, a média é altamente influenciada por esses valores altos e não expressa muito bem a melhor medida de tendência central desses dados, tendo valor maior do que a mediana e da moda.

Quando a distribuição dos dados não está na normal, a medida de tendência central mais representativa não é a média, mas sim a mediana.

Vamos verificar esses achados na prática, rodando os comandos abaixo no R.

```
#criando duas variáveis numéricas em um banco de dados:
set.seed(345)
cap9 <- data.frame(
  dados1 = rnorm(100, mean = 1, sd = 0.5),
  dados2 = rlnorm(100, meanlog=1, sdlog=0.5)
)

#você pode abrir o banco baixando o arquivo cap9.txt do Github

#Figura 9.4
#Verificando o formato da distribuição através do histograma
par(mfrow = c(1,2))
hist(cap9$dados1, main = 'Distribuição normal', xlab = 'Valores')
hist(cap9$dados2, main = "Distribuição fora da normal", xlab="Valores")

#criando a função "moda"
moda <- function(x){
  uniqx <- unique(x)
  uniqx[which.max(tabulate(match(x, uniqx)))]
}

#Verificando a diferença que tem a média da moda:
mean(cap9$dados1) - moda(cap9$dados1)
```

```
[1] 0.3544743

mean(cap9$dados2) - moda(cap9$dados2)
[1] 1.651839
```

O que fizemos aqui foi criar duas variáveis em um banco: uma na distribuição normal usando a função rnorm() e outra na distribuição log-normal (que não é a normal que conhecemos) usando a função rlnorm(). Depois, exploramos o histograma de cada uma delas, possível de serem vistos na Figura 9.4. Logo após, criamos a função global chamada "moda" e fizemos a diferença entre a média e a moda de cada banco de dados. Perceba que a diferença maior está em "dados2", ou seja, a média está mais distante da moda em "dados2" do que "dados1".

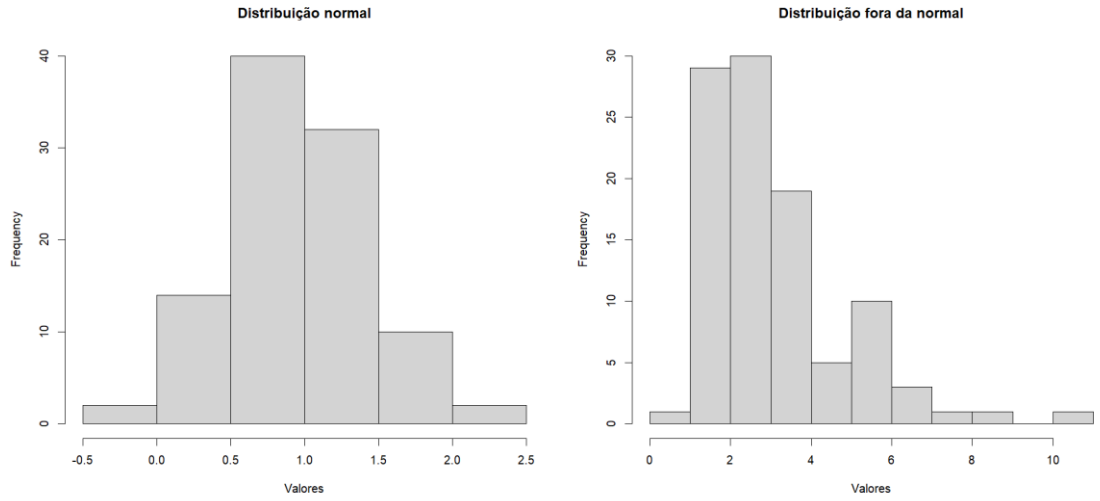


Figura 9.4. Comparação visual entre distribuições normal e não normal.

Rodando o comando abaixo, criaremos um *data frame* simples na intenção de ser uma pequena tabelinha, para entendemos o que ocorre entre essas duas distribuições em relação à média, mediana, moda e valores mínimos e máximos.

```
#definindo um data frame com duas colunas
df <- data.frame(
  dados1 = NA,
  dados2 = NA
)

#medidas
df[1,1] <- mean(cap9$dados1)
df[1,2] <- mean(cap9$dados2)
df[2,1] <- median(cap9$dados1)
df[2,2] <- median(cap9$dados2)
df[3,1] <- moda(cap9$dados1)
df[3,2] <- moda(cap9$dados2)
df[4,1] <- min(cap9$dados1)
df[4,2] <- min(cap9$dados2)
```

```
df[5,1] <- max(cap9$dados1)
df[5,2] <- max(cap9$dados2)

#mudando os nomes das linhas de df
rownames(df) <- c('Média', 'Mediana', 'Moda', 'Mínimo', 'Máximo')

df
            dados1     dados2
Média    0.9620202  3.0258932
Mediana  0.8739678  2.4870038
Moda     0.6075459  1.3740541
Mínimo  -0.2910301  0.9112566
Máximo   2.4091394 10.6713394
```

O que é importante perceber aqui é que a média, a mediana e a moda de "dados1" são bem próximas, o que característico de uma distribuição normal. Por outro lado, em "dados2" essas medidas são mais diferentes entre si, sendo a média maior que a mediana, e a mediana maior do que a moda, o que é característico de uma distribuição fora da normal com assimetria positiva (rever Figura 9.4 – gráfico direito).

9.1 TESTES DA NORMALIDADE DOS DADOS

Para sabermos estatisticamente se a distribuição dos dados se encaixa na distribuição normal ou não, nós utilizamos testes de normalidade. Existem diversos testes para isso: Jarque-Beta, D'Agostino-Pearson, Shapiro-Wilk, Kolmogorov-Smirnov (K-S), teste Lilliefors que é uma correção do K-S, teste Anderson-Darling, Carmer-von Mises e alguns outros.

Neste final de capítulo falaremos e demonstraremos a usabilidade de três deles, que são os mais utilizados: D'Agostino, Shapiro-Wilk e Kolmogorov-Smirnov. A diferença entre eles é basicamente o modelo matemático por detrás da análise. Grosseiramente falando, o teste D'Agostino avalia como se mostra o formato de sino da distribuição. Está simétrico? Está muito achatado ou se espalha demais no gráfico? E assim ele avalia se determinada distribuição dos dados está próxima ou não da distribuição normal. O teste Kolmogorov-Smirnov se baseia na distância entre distribuições: será que a distribuição dos dados está "distante" da distribuição normal? O Shapiro-Wilk utiliza uma estatística de teste que compara a distribuição empírica dos dados com a distribuição normal. Há uma discussão grande de qual teste de normalidade é o melhor. Em termos gerais, o Shapiro-Wilk é o melhor teste quando se tem uma amostragem pequena (menor que 50 observações).

Como vimos um pouco mais acima, amostras pequenas interfere muito no surgimento ou assemelhar-se com a distribuição normal. Isso não interfere só nesse quesito, mas em vários outros testes estatísticos. Evitar amostragens menores que 20 observações já é um ótimo começo.

Em termos gerais, quando realizamos um teste de normalidade as hipóteses em jogo são:

H_0: a distribuição dos dados está na normal.

H_1: a distribuição dos dados não está na normal.

Consideramos costumeiramente o nível de significância de 5%, temos que:

- Se o p-valor $\leq 0{,}05$, rejeitamos $H_0 \rightarrow$ a distribuição está fora da normal.
- Se o p-valor $> 0{,}05$, aceitar $H_0 \rightarrow$ a distribuição está na normal.

Portanto, vamos praticar no R, usando funções de um pacote chamado fBasics. Para o teste de normalidade Kolmorov-Smirnov, podemos usar a função ksnormTest(). Para Shapiro-Wilk, podemos utilizar shapiroTest(). E, para o D'Agostino, usamos dagoTest(). Se você ainda não tem o pacote fBasics, vamos instalá-lo. Vamos aproveitar dos dados do banco cap9 e testar as variáveis dados1 e dados2.

```
install.packages('fBasics')
library(fBasics)

d1_ks <- ksnormTest(cap9$dados1)
d1_sw <- shapiroTest(cap9$dados1)
d1_da <- dagoTest(cap9$dados1)
```

Explorando os resultados temos:

```
d1_ks

Title:
 Asymptotic one-sample Kolmogorov-Smirnov test

Test Results:
  STATISTIC:
    D: 0.5774
  P VALUE:
    Alternative Two-Sided: < 2.2e-16
    Alternative      Less: < 2.2e-16
    Alternative   Greater: 0.9799
```

```
d1_sw
```

```
Title:
 Shapiro - Wilk Normality Test

Test Results:
  STATISTIC:
    W: 0.9865
  P VALUE:
    0.407
```

```
d1_da

Title:
 D'Agostino Normality Test

Test Results:
  STATISTIC:
    Chi2 | Omnibus: 1.6275
    Z3  | Skewness: 1.2409
    Z4  | Kurtosis: 0.2959
  P VALUE:
    Omnibus  Test: 0.4432
    Skewness Test: 0.2146
    Kurtosis Test: 0.7673
```

Entre os três testes ocorreu uma coisa interessante. Para o teste Kolmogorov-Smirnov, a distribuição dos dados1 está fora da normal, pois consideramos o p-valor em "Alternative Two-Sided", o qual é menor do que 0,05. Para o teste Shapiro-Wilk o p-valor não foi significativo, então, ele considera que a distribuição está na normal. Para o teste D'Agostino, ele avalia três parâmetros de resultado: (1) qui-quadrado, para saber se os dados esperados são parecidos com os dados observados; (2) simetria, para saber basicamente se o gráfico da distribuição é similar a um sino ou ele tem assimetria negativa ou positiva; (3) curtose, para verificar o achatamento da distribuição, verificando se os dados estão próximos da média ou não. No caso dos p-valores retornados pelo teste D'Agostino, todos eles são maiores que 0,05, indicando normalidade dos dados.

Aqui temos 2 contra 1. Para fechar a dúvida se está ou não na distribuição os dados1, basta rever o gráfico da Figura 9.4, à esquerda, que mostra claramente a conformação de sino. Então dados1 está na distribuição normal.

Agora vamos verificar como anda a distribuição em dados2 de cap9.

```
d2_ks <- ksnormTest(cap9$dados2)
d2_sw <- shapiroTest(cap9$dados2)
d2_da <- dagoTest(cap9$dados2)
```

Vamos avaliar seus resultados.

```
d2_ks

Title:
 Asymptotic one-sample Kolmogorov-Smirnov test

Test Results:
  STATISTIC:
    D: 0.8385
  P VALUE:
    Alternative Two-Sided: < 2.2e-16
    Alternative      Less: < 2.2e-16
    Alternative   Greater: 1
```

```
d2_sw

Title:
 Shapiro - Wilk Normality Test

Test Results:
  STATISTIC:
    W: 0.8661
  P VALUE:
    4.899e-08
```

```
d2_da

Title:
 D'Agostino Normality Test

Test Results:
  STATISTIC:
    Chi2 | Omnibus: 39.1773
    Z3  | Skewness: 5.1632
    Z4  | Kurtosis: 3.5381
  P VALUE:
    Omnibus  Test: 3.11e-09
    Skewness Test: 2.427e-07
    Kurtosis Test: 0.0004029
```

Perceba que os p-valores gerados agora, seja para K-S (em *Alternative Two-Sided*), Shapiro-Wilk ou D'Agostino, todos eles são muito menores do que 0,05, ou seja, significativos. Isso sugere que devemos rejeitar a hipótese de que os dados estão na distribuição normal. Portanto, a distribuição de dados2 está fora da normal.

9.2 Q-Q PLOT

O Q-Q plot é uma alternativa gráfica muito útil para avaliar se uma determinada amostra de dados segue uma distribuição normal ou não. No Q-Q plot, se os pontos no gráfico se aproximarem de uma linha diagonal (que será a referência para a distribuição normal), isso sugere que a amostra segue uma distribuição normal. Se os pontos se afastarem significativamente da linha diagonal, isso sugere que a amostra não segue a distribuição normal. Para fixar resumidamente, se os pontos formarem uma linha reta, então a distribuição dos dados é considerada normal. Se a linha formada for curvada para cima e/ou para baixo, consideramos fora da normal.

Na prática, vamos continuar usando cap9 e os valores de dados1 e dados2 do subcapítulo acima. Vamos abrir as bibliotecas tidyverse e patchwork para criar os gráficos (mas não se atenha demais ao código deles por agora, pois aprenderemos cada ponto no capítulo sobre gráficos do R). O último pacote servirá apenas para agrupar os dois gráficos lado a lado e se você ainda não o tiver, instale-o no R. A função geom_qq() criará o Q-Q plot em si, e a geom_qq_line() irá colocar no gráfico a linha transversal que é a nossa referência da normal.

```
#Figura 9.5
library(tidyverse)
g1 <- cap9 %>% ggplot(aes(sample = dados1)) +
  geom_qq(size = 3) +
  geom_qq_line()

g2 <- cap9 %>% ggplot(aes(sample = dados2)) +
  geom_qq(size = 3) +
  geom_qq_line()


library(patchwork)
g1 + g2
```

A imagem gerada será igual à Figura 9.5, a seguir. Perceba que no gráfico à esquerda os pontos estão bem próximos à linha de referência. Quando temos uma distribuição normal dos nossos dados, é isso que queremos ver. Já no gráfico à direita, perceba que os pontos escapam da reta de referência com mais frequência. Nesse segundo caso, os dados não estão na normal.

Tem gente que tem pavor quando a distribuição dos dados se encontra fora da normal e acha que todo o trabalho de coleta de dados está perdido. Isso não é verdade. Entenda que estar na distribuição normal ou não jamais se equivale a ser uma coisa boa ou ruim, mas sim que o tratar dos dados e a forma de analisa-los será diferente.

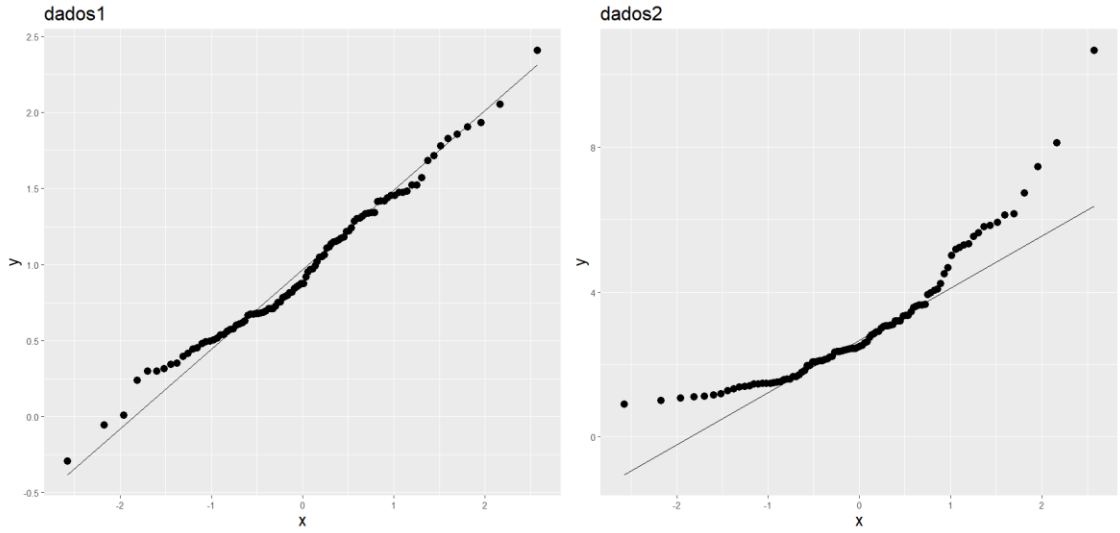


Figura 9.5. Q-Q plots das variáveis dados1 e dados2 do banco cap9.

Quando a normalidade dos dados é verdadeira, utilizamos teste estatísticos chamados testes paramétricos.
Quando a normalidade não é encontrada nos dados, usamos testes não paramétricos.

Capítulo

10

Correlação

Uma das grandes máximas da bioestatística é tentarmos descobrir relações entre variáveis: aumento do IMC e risco de doenças cardíacas, baixa de hemoglobina e anemias, comer alimentos saudáveis e surgimento de certos tipos de câncer, idade e memória, quantidade de sono e desempenho cognitivo, enfim. Nós corremos atrás dessas relações para evitarmos, muitas das vezes, fatores de risco que nos expunham a doenças e outros males da vida.

Apesar de que linguisticamente os termos correlação e associação sejam quase sinônimos, na bioestatística não são. Correlação e associação são testes estatísticos que no fundo buscam avaliar relação entre variáveis, mas que são calculados de formas completamente diferentes.

Neste capítulo, nos preocuparemos em entender como são feitos e o modo de interpretar os testes mais comuns de relação entre variáveis numéricas. Esses são os testes de correlação. Já as associações estatísticas, nós nos preocupamos na relação entre variáveis qualitativas, que veremos em capítulos mais à frente.

Algumas análises são melhor compreendidas com dispositivos visuais. Para entendermos esse início sobre as correlações estatísticas, temos um tipo de gráfico muito útil chamado de diagrama de dispersão. Nada mais, nada menos, é do que um gráfico de pontos para o R. Sendo assim, vamos importar um pequeno *data frame* que contenha variáveis numéricas chamado cap10. Ele terá os seguintes dados:

```
cap10 <- data.frame(
  imc = c(30.2, 23.2, 23.3, 24.2, 25.1, 25.2, 25.3, 26.3, 26.9,27.4,28.0, 32.4, 29.0,
32.7, 29.8),
  PAS = c(165, 110, 114, 117, 122, 122, 121, 125, 121, 127, 128, 189, 140, 176, 146),
  HDL = c(25,48,56,61,48,52,38,35,31,34,25,16,31,25,35),
  testo = c(635.1, 564.6, 204.9, 487.9, 911.9, 365.9, 668.1, 838.9, 623.9, 324.4,
697.4, 734.1, 448.7, 163.1, 967.5)
)

  #ou

library(rio)

cap10 <- import('caminho_da_pasta/cap10.txt ')
```

Esse banco há 15 observações de dados fictícios de índice de massa corpórea (variável imc), pressão arterial sistólica (PAS), dosagens séricas de HDL e dosagens séricas de testosterona total no sangue (testo). Não nos preocupemos com as unidades de cada variável no momento, mas saiba que isso é importante no processo de escrita científica. Por ora, a primeira coisa que podemos fazer é avaliar graficamente como estão as relações entre cada variável. Vamos nos ater às relações de IMC com as demais.

Para fazermos os diagramas de dispersão, vamos construir gráficos de pontos, como dito mais acima. O diagrama se baseia em eixos cartesianos: X e Y. Os pontos que o gráfico contém é o valor da variável do eixo Y correspondente ao valor da variável no eixo X para cada observação (cada paciente, no caso). Melhor que entender esta frase é demonstra-la. Vamos criar um diagrama de dispersão colocando a variável imc no eixo X e a variável PAS no eixo Y.

```
#Figura 10.1 (esquerda)
plot(x = cap10$imc, y = cap10$PAS)
```

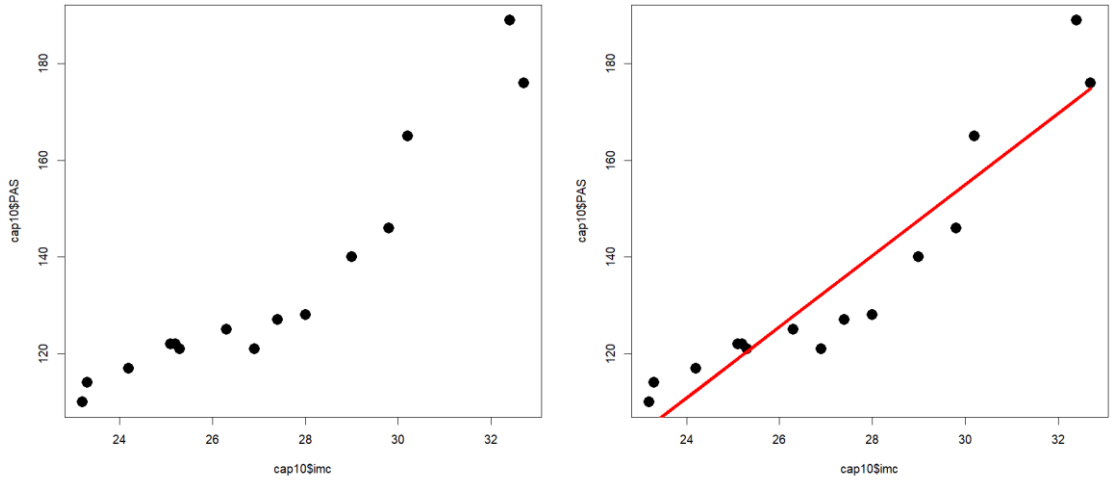


Figura 10.1. Diagrama de dispersão entre IMC e pressão arterial sistólica.

Analisando o comportamento do diagrama de dispersão da Figura 10.1, à esquerda, podemos dizer que enquanto se aumenta o IMC, a pressão sistólica tende a aumentar concomitantemente. Então, graficamente falando, existe uma tendência de que quanto maior o IMC, maior será a PAS. Perceba que no gráfico à direita se mostra uma linha vermelha de tendência desses dados, o que confirma graficamente a nossa tese. Falando de forma coloquial, ela vai "subindo" no gráfico. Enquanto X "sobe", Y "sobe" junto. Isso é comum quando temos uma **correlação positiva** entre as variáveis.

Vejamos agora o comportamento da relação entre o IMC e o HDL.

```
#Figura 10.2
plot(x = cap10$imc, y = cap10$HDL
```

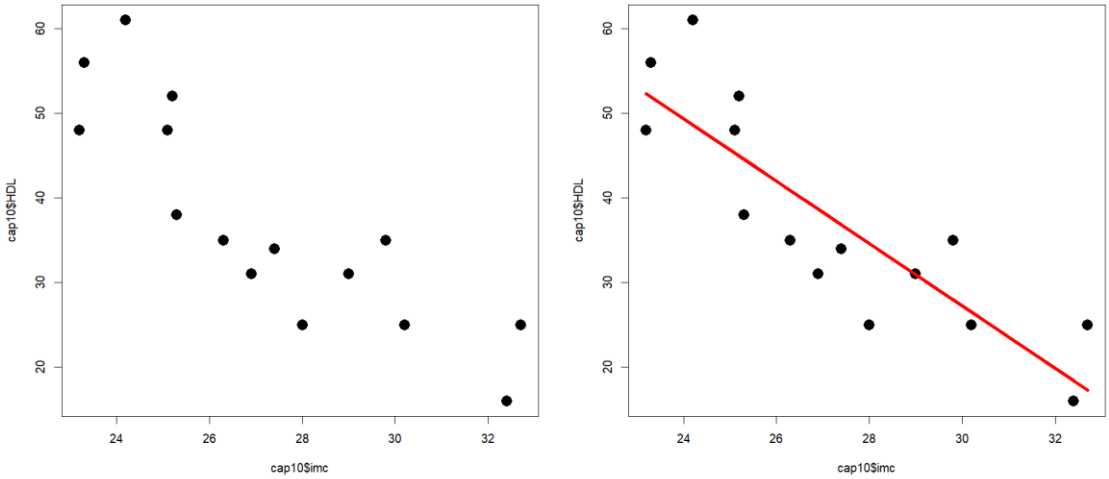


Figura 10.2. Diagrama de dispersão entre IMC e o HDL sérico.

Perceba que o comportamento é inverso à relação IMC vs. PAS. Aqui, enquanto se aumenta o IMC, o HDL tende a diminuir, tendo uma relação antagônica. Observe a linha de tendência vermelha na Figura 10.2 à direita que ela obliquamente "desce". Coloquialmente falando, enquanto X "sobe", Y "desce". Isso é comum quando temos uma **correlação negativa** entre as variáveis.

Agora, vamos avaliar o comportamento gráfico da relação entre IMC e os níveis de testosterona (texto) em nossa amostragem. Para isso, vamos construir o diagrama abaixo.

```
#Figura 10.3
plot(x = cap10$imc, y = cap10$testo)
```

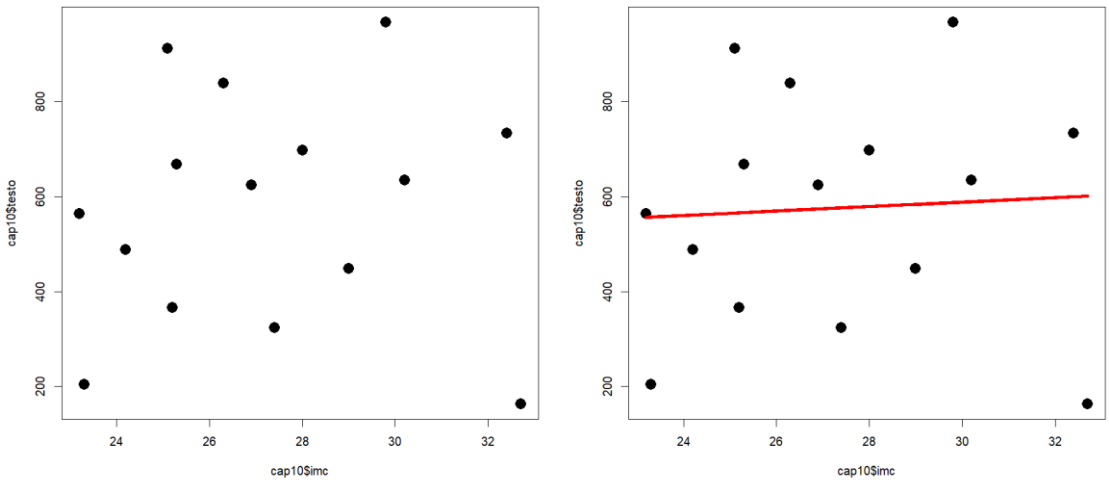


Figura 10.3. Diagrama de dispersão entre IMC e os níveis de testosterona total sérica.

Perceba que na Figura 10.3, à esquerda, os pontos estão muito dispersos no gráfico. Quando isso ocorre, é um grande indício de que as variáveis X e Y não têm relação significativa entre elas. No gráfico à direta, perceba que a linha vermelha de tendência aparece "deitada", na horizontal. Outro indício de que não há correlação entre as variáveis em análise.

- Correlação positiva: quando pares de valores, X e Y crescem conjuntamente no gráfico.
- Correlação negativa: quando os valores de X crescem, enquanto os valores de Y respectivos decrescem.
- Correlação nula: quando pares de valores, X e Y, se mostram bem dispersos no diagrama de dispersão.

Já temos conhecimento suficiente para termos visão crítica, visualmente falando, se uma variável está correlacionada a outra. Contudo, só podemos confirmar essa hipótese testando o que chamamos de coeficiente de correlação. Existem dois testes básicos que você precisa conhecer: (1) o coeficiente de correlação de Pearson (*r*); e, (2) o coeficiente de correlação de Spearman (ρ – letra grega rô). Você utilizará um ou o outro, nunca os dois ao mesmo tempo.

Mas como pode isso? Bom, depende de como estão distribuídos os dados. Caso os dados, das duas variáveis estiverem na distribuição normal, então utilizaremos o coeficiente de correlação de Pearson (que é o mais conhecido). Caso contrário, se a distribuição dos dados das duas, ou pelo menos uma das variáveis, estiverem fora da normal, utilizaremos o coeficiente de correlação de Spearman. Veja o esquema básico na Figura 10.4 logo abaixo.

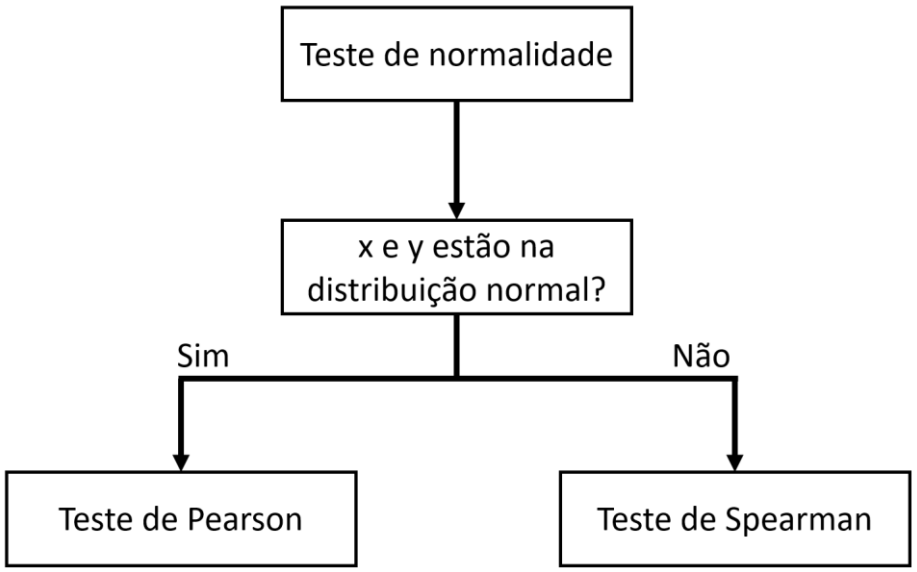


Figura 10.4. Esquema para saber qual coeficiente de correlação utilizar.

O coeficiente de correlação de Pearson pode ser calculado pela fórmula abaixo:

$$r = \frac{\sum XY - \frac{(\sum X)(\sum Y)}{n}}{\sqrt{\left[\sum X^2 - \frac{(\sum X)^2}{n}\right]\left[\sum Y^2 - \frac{(\sum Y)^2}{n}\right]}}$$

Onde X e Y são os dados conjuntos de cada observação. Já a fórmula para calcular o coeficiente de correlação de Spearman rô é:

$$\rho = 1 - \frac{6\sum d^2}{n^3 - n}$$

A maior diferença aqui é que esse coeficiente de Spearman não é calculado utilizando os dados diretamente. Ele utiliza-se dos postos (d). Esses postos são uma forma de transformar os dados originais em uma escala ordinal, ou seja, uma escala na qual os dados são ordenados com base em sua classificação em relação aos demais. Não faz parte do escopo deste livro adentrar em mais detalhes sobre como são calculados matematicamente cada coeficiente, uma vez que a proposta aqui é a prática e a interpretação dos resultados gerados pelo R.

Ambos os coeficientes têm a mesma interpretação. Eles variam entre -1 e 1. Esses valores extremos indicam correlação perfeita, contudo: (a) -1 indica **correlação negativa** perfeita; (b) +1 indica **correlação positiva** perfeita. Perceba que o intervalo passa pelo zero e todo coeficiente que é igual a zero ou muito próximo indica que não há correlação entre as variáveis.

Isso tem tamanha importância que há um teste estatístico próprio cuja hipótese nula (H_0) afirma que o coeficiente de correlação é igual a zero. Se o p-valor desse teste for significativo, ou seja, menor do que 0,05, então admitimos que o coeficiente de correlação é diferente de zero. Exemplo, se *r* = 0,74 e seu p-valor = 0,0002, então esse coeficiente de correlação é realmente 0,74; uma correlação positiva.

Se o p-valor não for significativo, maior a 0,05, então esse coeficiente de correlação estatisticamente é considerado igual a zero, não importando o seu real valor calculado, não havendo correlação entre as variáveis. Exemplificando, se *r* = 0,15 e seu p-valor = 0,84, então o coeficiente de correlação é considerado zero. Se você achou um pouco confuso o p-valor ter entrado aqui, relei só mais uma vez e junto com a prática mais abaixo você entenderá perfeitamente.

Schober *et al* (2018) resumiu em uma tabela intervalos classificatórios para os valores dos coeficientes de correlação. Na Tabela 10.1 você pode usar como referência para interpretar as

correlações entre as variáveis em jogo. Apesar da tabela resumir valores positivos, você também deve considerar as mesmas classificações para correlações negativas. Exemplo, um coeficiente de correlação igual a -0,85 é uma correlação negativa forte.

Se você procurar, certamente vai encontrar outros autores que estabeleceram de forma diferente os intervalos e suas classificações, contudo, você perceberá ao comparar que são muito parecidos. A chave aqui é citar o autor o qual você se baseou e dê importância ao p-valor. Caso esse seja muito significativo (p-valor < 0,01), maior será a probabilidade desse coeficiente ser diferente de zero, maior será a correlação entre as variáveis em estudo.

Tabela 10.1. Abordagem interpretar um coeficiente de correlação.

Magnitude absoluta do coeficiente de correlação observado	Interpretação
0,00 – 0,10	Correlação insignificante
0,11 – 0,39	Correlação fraca
0,40 – 0,69	Correlação moderada
0,70 – 0,89	Correlação forte
0,90 – 1,00	Correlação muito forte

Fonte: Schober *et al.*, 2018.

No R base existem duas funções para calcularmos a correlação. Uma é a cor() e a outra é a cor.test(). Vamos testar as duas, contudo, vou adiantar que a função cor() retorna apenas o valor do coeficiente, o que é um resultado pobre. Para tirarmos conclusões mais assertivas do que está ocorrendo, precisamos de mais detalhes, os quais a função cor.test() é melhor para isso.

Em nossos exemplos, continuaremos a usar o banco cap10. Vamos nos certificar se há correlação ou não entre as variáveis IMC e pressão arterial sistólica, primeiramente. Mas antes disso, lembre-se: precisamos testar sempre a normalidade dos dados para saber qual o teste de correlação adequada a ser aplicado. Vamos lá, passo a passo.

```
shapiro.test(cap10$imc)    #testando a normalidade do IMC

      Shapiro-Wilk normality test
data:  cap10$imc
W = 0.9443, p-value = 0.4396

shapiro.test(cap10$PAS)   #testando a normalidade da pressão arterial

      Shapiro-Wilk normality test
data:  cap10$PAS
W = 0.82602, p-value = 0.008091
```

Perceba que a variável PAS não está na normal, pois o p-valor para o teste Shapiro-Wilk foi significativo. Dessa forma, devemos utilizar o teste de correlação de Spearman (nosso teste de correlação não paramétrico). Para isso, precisamos modificar o parâmetro "method" igual a "spearman", como segue:

```
cor(cap10$imc,cap10$PAS, method = 'spearman')
[1] 0.9534899


cor.test(cap10$imc,cap10$PAS, method = 'spearman')

      Spearman's rank correlation rho

data:  cap10$imc and cap10$PAS
S = 26.046, p-value = 3.835e-08
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.9534899
```

Veja que utilizamos as duas funções: cor() e cor.test(). Perceba também que os valores de rô são exatamente iguais (0,9534899 – ou arredondando 0,95), mas o produto gerado por cada função é diferente. A função cor() não retorna o p-valor associado ao teste de hipótese que o coeficiente seja estatisticamente igual a zero, já cor.test() traz esse teste. Esse p-valor de $3{,}8 \times 10^{-8}$ é significativo (porque foi menor do que 0,05). Portanto, o coeficiente não é estatisticamente igual a zero. Em suma, há evidencias suficientes para dizermos que há correlação positiva muito forte entre IMC e a pressão sistólica.

Vamos testar agora a correlação entre IMC e dosagens de HDL (lipoproteína de alta densidade) sérica. Já sabemos que a variável IMC da nossa amostragem está na distribuição normal, basta agora testar a normalidade da variável HDL.

```
shapiro.test(cap10$HDL)

      Shapiro-Wilk normality test

data:  cap10$HDL
W = 0.95145, p-value = 0.5475
```

Percebemos que ambas IMC e HDL estão na distribuição normal. Então, devemos testar a correlação entre elas usando o método de Pearson (o nosso teste de correlação paramétrico). Modifique "method" para "pearson" ou meramente omita esse parâmetro, uma vez que o método de Pearson é o padrão das funções de teste de correlação. Vamos ser diretos e usar apenas a cor.test().

```
cor.test(cap10$imc,cap10$HDL, method = 'pearson')
```

```
      Pearson's product-moment correlation

data:  cap10$imc and cap10$HDL
t = -6.1905, df = 13, p-value = 3.268e-05
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.9540609 -0.6313086
sample estimates:
       cor
-0.8641189
```

Como resultado, vemos que há correlação entre IMC e HDL (p-valor ≤ 0,05). Essa correlação é forte e negativa (r = -0,86). Isso significa que à medida que aumenta os valores de IMC, temos a tendência de que HDL vá reduzindo. A estatística fica por aqui, por enquanto, mas você precisa ir além. Consegue perceber que ter HDL baixo é um fator de risco para ter obesidade? Claro que existem outros testes estatísticos que nos ajudarão a termos mais certeza dessa afirmação, porém, só o teste de correlação já nos dá um *insight* muito importante nessa jornada.

Para finalizarmos nossos exemplos, precisamos conferir a correlação entre IMC e a dosagem de testosterona sanguínea.

```
shapiro.test(cap10$testo)

      Shapiro-Wilk normality test
data:  cap10$testo
W = 0.97304, p-value = 0.9002
```

Vemos que as dosagens de testosterona estão na distribuição normal, bem como as de IMC. Vamos seguir usando Pearson.

```
cor.test(cap10$imc,cap10$testo)

      Pearson's product-moment correlation

data:  cap10$imc and ccap10$testo
t = 0.21231, df = 13, p-value = 0.8352
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.4675595  0.5543527
sample estimates:
       cor
0.05878256
```

Como resultado, podemos dizer que o coeficiente de correlação de Pearson gerado é estatisticamente igual a zero, não havendo correlação entre IMC e os níveis séricos de testosterona (r = 0,06; p-valor = 0,84).

Capítulo

11

Regressão

As regressões servem fundamentalmente para prever uma variável sabendo uma outra ou várias outras variáveis. Para isso, envolve nessas técnicas estatísticas mecanismos matemáticos que acabam retornando a relação entre elas. Estatisticamente falando, na maioria das situações, usamos análises de regressão para conhecermos, de uma outra forma que não seja a correlação puramente, a relação entre variáveis.

Existem dezenas de técnicas de regressão, contudo, vamos abordar aqui duas delas: a regressão linear e a logística. A partir dessas, há várias variações, penalizações, enfim, técnicas matemáticas para melhorar a predição e assim achar a melhor relação entre as variáveis em jogo e o desfecho.

Primeira coisa que devemos saber para seguirmos adiante é o tipo de resposta que queremos ao final. Se eu tenho uma variável Y e quero prever seu valor segundo uma variável X, então chamamos Y de variável resposta, ou predita, ou variável **dependente**, ou ainda de desfecho em algumas situações. A variável X, ou variáveis outras, chamamos de preditora ou variável **independente**. Por exemplo, desejamos prever o valor do IMC usando a circunferência da cintura. IMC é a variável dependente e a circunferência da cintura é a variável independente. Outro exemplo, se um químico está pesquisando a dilatação da gasolina, relacionando isso com temperatura, pressão atmosférica e umidade do ar. Pois bem, os valores de dilatação do combustível é a variável predita e as demais serão as preditoras.

Outro fator que devemos abordar é a classificação da regressão quanto o seu número de variáveis independentes. Se na minha relação eu tenho apenas uma variável preditora, então estamos falando de uma regressão simples (caso da relação IMC x circunferência da cintura). Se estamos diante de duas ou mais variáveis preditoras, então estamos diante de uma regressão múltipla (caso da dilatação da gasolina e as outras três preditoras: temperatura, pressão atmosférica e umidade do ar. Ao longo de cada subcapítulo, iremos dar pelo menos dois exemplos, um de regressão simples e outro de regressão múltipla.

A nível de curiosidade, a origem do termo "regressão" na estatística tem fundo biológico. Foi introduzido por Francis Galton (1822-1911). Ele usou a palavra "regressão" para descrever a tendência de uma característica herdada a "regressar" ou voltar à média de uma população. Galton conduziu vários estudos sobre a transmissão de traços físicos e mentais de pais para filhos, e descobriu que a maioria das características tende a se agrupar em torno da média da população. Ele

percebeu que, quando os pais têm uma característica extrema, seus filhos tendem a ter uma característica menos extrema (regrediu), ou seja, mais próxima da média.

11.1 REGRESSÃO LINEAR

No capítulo anterior, sobre correlação, vimos a distribuição bidimensional dos dados em um gráfico de dispersão. Foi mostrado em alguns gráficos uma linha de tendência vermelha entre os pontos espalhados no gráfico. Pois bem, essa reta que se ajusta o melhor possível a esses pontos é chamada de reta de regressão (veja Figura 10.1 a 10.3).

Certamente, no seu ensino médio, você se deparou com isso nas aulas de matemática, quando o professor abordou equações do primeiro grau e expôs essa famosíssima fórmula: y = ax + b. Nesse momento juvenil, pouco imaginaríamos que ela fosse tão útil na área da saúde. Mas, isso aqui é só o básico. Vamos aprofundar e explorar melhor os achados possíveis que essa metodologia retorna e compreender como podemos interpreta-los e gerar conhecimento.

Mas, por que regressão linear? Pelo fato de estarmos assumindo que os nossos dados se ajustam bem em uma linha de tendência, como foi mostrado nas Figuras 10.1 a Figura 10.3. Contudo, nem sempre a tendência é perfeitamente uma reta, a partir daí, surgem outros vários tipos de regressões que tentam se ajustar melhor a cada tendência.

11.1.1 Regressão linear simples

Algebricamente, a equação citada acima apresenta os coeficientes: *a* e *b*. Padronizou-se na estatística uma leve mudança, chamaremos aqui de betas. Ou seja, estatisticamente falando, a equação de regressão linear simples apresenta essa conformação:

$$\hat{y} = b_0 + b_1 x$$

Chegamos então à equação de regressão linear simples. Ela é chamada simples porque estamos avaliando uma variável em função de outro. Resumidamente, há apenas duas variáveis em jogo: uma predita (dependente) e uma preditora (independente). Na realidade, algebricamente nada muda, exceto a forma demonstrativa. O b_0 (ou beta zero) chamaremos de **intercepto** e é onde a reta intercepta o eixo cartesiano y. Já o b_1 (beta um) pode ser chamado meramente de **inclinação**, porque ele interfere na inclinação da reta. Não esquecendo de que y é a variável dependente ou resposta e x é a variável independente.

Para calcularmos os betas, temos as seguintes fórmulas:

Inclinação: $b_1 = r\frac{s_y}{s_x} = \frac{\sum XY - \frac{(\sum X)(\sum Y)}{n}}{\sum X - \frac{(\sum X)^2}{n}}$

Intercepto: $b_0 = \bar{y} - b_1\bar{x}$

O *r* é o coeficiente de correlação entre x e y; *S*x e *S*y são os desvios padrão respectivos; $\overline{y}$ é a média de y e $\overline{x}$ é a média de x. Ou seja, exatamente tudo o que já conhecemos ao longo da nossa jornada até aqui e você pode calcular à mão, se quiser.

Perceba que a correlação entre as variáveis está definindo a inclinação da reta. Com uma correlação positiva, a reta terá uma inclinação ascendente. Quando houver uma correlação negativa, a reta terá uma inclinação descendente. Sem correlação, a inclinação é ínfima, com a reta praticamente na horizontal.

Iremos praticar no R usando um banco de dados fictício de um produtor que desejava estimar o peso de carcaça de suínos (kg) segundo a sua idade (em meses). Chamaremos de "regsim" esse banco. Mas, antes de mais nada, é importante você saber que o que iremos fazer é criar um modelo. Um modelo matemático é uma representação abstrata e simplificada de um sistema ou fenômeno da vida real, utilizando ferramentas matemáticas e computacionais para descrever e analisar suas propriedades e comportamentos. Assim, esses modelos podem ser usados para prever resultados futuros, entender melhor o funcionamento do fenômeno em questão, testar hipóteses e tomar decisões com base no modelo criado. Dessa forma, qualquer análise que fizermos daqui para frente, que envolva a criação de uma equação matemática, estamos criando um modelo. A regressão nada mais é do que um modelo preditivo.

Com a função lm() iremos criar esse tal modelo. L e M vêm do inglês *linear models*. Nesse momento, você precisa conhecer o termo "fórmula" que é utilizada dentro de algumas funções do R. Fórmulas do R não são as fórmulas matemáticas. São na realidade relações escritas em código. A primeira coisa é identificarmos a variável dependente, que será o nosso y: peso de carcaça. A idade será a variável independente, ou preditora. Ciente disso, podemos escrever genericamente a seguinte fórmula: carcaça ~ idade. O til limita à esquerda a variável dependente, e à direta a variável independente. Vamos praticar e mais na frente interpretaremos os resultados da regressão simples que serão mostrados pela função summary().

```
#criando o banco
regsim <- data.frame(
  idade = c(6,7,8,7,10,9,12,11,14,15,6,11,14),
  carc = c(89,88,90,98,89,112,110,124,130,136,78,100,127)
)

#criação do modelo
result_rls <- lm(carc ~ idade, data = regsim)

#observando os resultados
summary(result_rls)

Call:
lm(formula = carc ~ idade, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-16.4615  -5.9700   0.0469   5.0300  13.1656

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  51.7327     8.8157   5.868 0.000108 ***
idade         5.3729     0.8441   6.365 5.33e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.169 on 11 degrees of freedom
Multiple R-squared:  0.7865,    Adjusted R-squared:  0.7671
F-statistic: 40.52 on 1 and 11 DF,  p-value: 5.331e-05
```

A primeira informação retornada por summary() não chega ser um resultado em si, é o “Call”, que mostra a fórmula usada para a criação do modelo de regressão. É um ponto de checagem. Depois, em “Residuals”, o R retorna a análise de resíduos, que nada mais são do que a diferença do dado real do que foi predito. Essa predição existe um erro intrínseco e dificilmente será exatamente o mesmo valor do dado original. Para entender melhor o que é tudo isso, veja a Figura 11.1: em (A) está exposto o diagrama de dispersão, ou gráfico de pontos; (B) está a nossa reta de regressão, e perceba que ela segue bem a tendência dos dados; em (C) veja que há uma distância de cada ponto à reta. Essa distância são os resíduos. É o quanto os dados reais fogem da nossa reta de regressão (da nossa previsão). Quanto maior forem esses resíduos, mais dispersos os dados estarão da reta, e isso não é muito interessante, pois o nosso modelo não estaria prevendo adequadamente nossos dados reais.

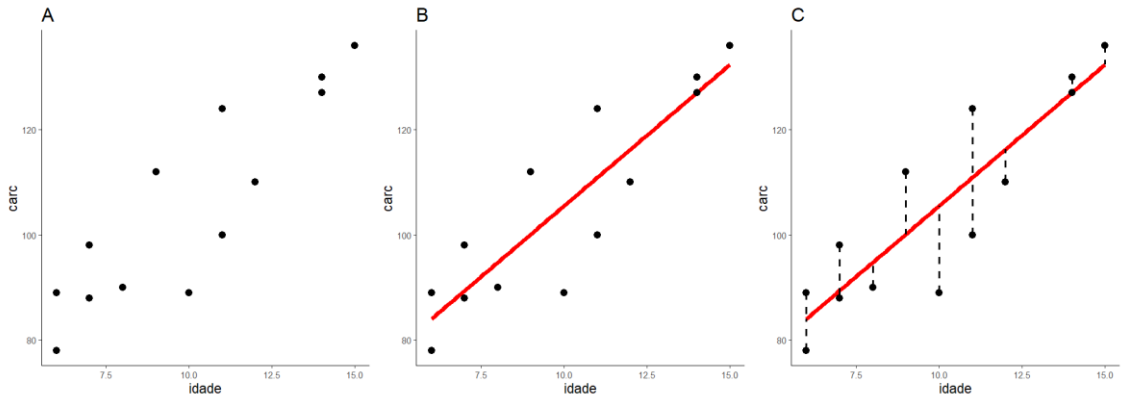


Figura 11.1. Reta de regressão e seus resíduos
A – Diagrama de dispersão; B – Reta de regressão; C – Análise de resíduos.

Para construir a Figura 11.1, eu utilizei os códigos a seguir, mas não se atenha muito a eles nesse momento, pois falaremos bastante sobre gráficos em um capítulo específico.

```
#prevendo valores de carcaça
regsim$previsao <- predict(result_rls)

#calculando o valor real pelo previsto (resíduos)
regsim$residuos <- regsim$carc - regsim$previsao

#gráfico de resíduos
library(tidyverse)
g1 <- ggplot(data = regsim, aes(x = idade, y = carc)) +
  geom_point(size = 3) +
  labs(title = 'A') +
  theme_classic() +
  theme(axis.title = element_text(size = 14),
        title = element_text(size = 16))

g2 <- ggplot(data = regsim, aes(x = idade, y = carc)) +
  geom_smooth(method = "lm", se = FALSE, color = "red", linewidth = 2) +
  geom_point(size = 3) +
  labs(title = 'B') +
  theme_classic() +
  theme(axis.title = element_text(size = 14),
        title = element_text(size = 16))

g3 <- ggplot(data = regsim, aes(x = idade, y = carc)) +
  geom_smooth(method = "lm", se = FALSE, color = "red", linewidth = 2) +
  geom_segment(aes(xend = idade, yend = previsao), color = "black", linetype =
"dashed", linewidth = 1) +
  geom_point(size = 3) +
  labs(title = 'C') +
  theme_classic()+
  theme(axis.title = element_text(size = 14),
        title = element_text(size = 16))
```

```
#conbinar os gráficos
library(patchwork)
g1 + g2 + g3
```

Em suma, a análise de resíduos é uma técnica utilizada para avaliar a adequação do modelo escolhido para os dados observados. E aqui eu deixo mais uma informação valiosa: é interessante que os resíduos estejam na distribuição normal. Se assim o for, indica que o modelo escolhido capturou a essência da relação entre as variáveis e que não há viés na previsão do modelo.

Em uma análise de dados, para uma publicação de um artigo científico, normalmente os autores não expõem a análise dos resíduos, mas o pesquisador deve se preocupar com ela.

Continuando a análise dos nossos resultados, temos "Coefficients". A coluna "Estimate" detém os valores dos coeficientes, ou seja, os betas da regressão. Sendo assim, podemos construir a nossa fórmula matemática de regressão linear simples da seguinte maneira:

$$y = 51{,}7327 + 5{,}3729\,x$$

Com essa fórmula podemos prever o peso em quilograma de carcaça de qualquer animal da fazendo, bastando saber a idade em meses dele. Se soubermos que há um suíno de 7,5 meses (ou seja, 7 meses e duas semanas), espera-se que ele tenha 92,03 kg. Você pode fazer na calculadora, ou basta usar a função predict() para predizer qualquer valor que queira tendo o modelo construído. Veja:

```
#predizendo um valor específico
predict(result_rls, newdata = data.frame(idade = 7.5))
       1
92.02934
```

Usando predict(), você precisa ter alguns cuidados, porque ela só aceita prever dados novos que estão em um objeto do tipo *data frame*. Além disso, o nome da variável dentro desse novo *data frame* precisa ser exatamente igual à variável independente que criamos o modelo lá com lm(). Fora isso, o céu é o limite.

Voltando a "Coefficients", há um resultado muito importante, o p-valor associado à variável preditora. Na realidade esses p-valores vêm de um teste de hipótese que avalia se os betas da regressão são estatisticamente iguais a zero. Então, o teste de hipótese para cada beta pode ser escrito assim:

H_0: $beta_i$ é igual a zero.

H_1: $beta_i$ é diferente de zero.

onde $_i$ é qualquer beta do modelo. Se o beta for estatisticamente igual a zero, perceba que a variável respectiva não exerce influência sobre o desfecho. Se em $y = b_0 + b_1.x$, o b_1 for zero, teríamos na equação apenas $y = b_0$. Então, qualquer que fosse o valor da variável x não teria efeito nenhum sobre y.

Se o p-valor for significativo ($P \leq 0,05$), então o seu beta respectivo é diferente de zero, e é o que a maioria dos analistas desejam na maioria das análises em bioestatística. Mas, por que isso?

Perceba na função acima que quanto maior for o b_1, em módulo, maior será a influência dessa variável preditora em relação ao desfecho. E, outra coisa interessante é que os valores de beta podem ser tanto valores positivos quanto negativos. Quando positivo, quer dizer que a variável preditora x tem relação positiva com a variável predita y, ou seja, quando uma aumenta a outra aumenta conjuntamente. Quando o beta é negativo, quer dizer que a relação entre elas é antagônica. Quando o beta é estatisticamente igual a zero ($P > 0,05$), então não há associação entre elas.

Ao lado do p-valor de cada beta respectivo visto ainda nos resultados, há uma simbologia comum no ambiente científico. Usamos quantidades de asteriscos quanto mais significante for os p-valores. Normalmente não passamos de três (***), o que significa que o p-valor é menor do que 0,001.

O "Residual standard error" é a estimativa do desvio padrão dos resíduos/erros de predição do modelo; geralmente não aprofundamos muito aqui. O "Multiple R-squared" é o nosso R^2 (lê-se erre quadrado), ou **coeficiente de determinação**, que é uma medida que quantifica a variância na variável dependente que é explicada pela variável independente. Quanto mais próximo de 1, melhor é o ajuste do modelo aos dados. Esse resultado é interessante a ser aprofundado, porque ele é exatamente o coeficiente de correlação entre as duas variáveis elevado ao quadrado, quando estamos falando de regressão linear simples. Mas na realidade, em essência, o R^2 é calculado segundo fundamentos matemáticos que consideram a variação entre o que foi predito e a média de y sobre a variação total. Por ora, com nosso modelo, foi retornado 0,7865, vamos testar se o R^2 bate exatamente com o coeficiente de correlação entre as variáveis?

```
cor(regsim$idade, regsim$carc)^2
[1] 0.7864755
```

É exatamente o mesmo número, apenas arredondado para quatro casa decimais. Podemos interpretar esse resultado da seguinte forma: 78,65% da variação total de carcaça podem ser explicados pela idade do animal. Os outros 21,35% não podem ser explicados pela idade, mas sim por

algum outro fator e/ou variação aleatória. No contexto, existem outros fatores que interferem no peso de carcaça, como o tipo de alimentação, qualidade da ração, qualidade de vida do animal, entre diversos outros, mas nada disso entrou em jogo para a construção do modelo de regressão linear, nesse momento.

Ainda há o "Adjusted R-squared", que é o R^2 ajustado, ajustado pelo número de variáveis independentes no modelo. Por conta disso, ele passa a ser útil ao criar modelos de regressão múltipla.

Finalizando a interpretação dos resultados, a estatística F é outra métrica que é usada para testar a significância global do modelo. É calculada com base na soma dos quadrados dos resíduos. O seu p-valor associado é o teste da hipótese nula de que todos os coeficientes de regressão são iguais a zero. Valores de P baixos indicam que o modelo como um todo é estatisticamente significativo (é o que todo analista deseja encontrar na vida). O resultado retornado em nosso exemplo foi p-value = 5,331e-05, ou seja, 0,00005331; significativo.

11.1.2 Regressão linear múltipla

A regressão múltipla é uma técnica estatística utilizada para modelar a relação entre uma variável dependente e duas ou mais variáveis independentes. É, portanto, uma extensão da análise de regressão simples. A regressão múltipla permite examinar como várias variáveis independentes podem influenciar a variável dependente simultaneamente, levando em consideração suas interações e efeitos individuais. Por isso, as regressões múltiplas são mais utilizadas na área biológica, porque a vida é comumente multifatorial e complexa.

A equação da regressão múltipla é:

$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + ... + b_kx_k$$

ou seja, beta zero continua sendo o intercepto e os demais betas são os coeficientes respectivos de cada variável independente em estudo. Então, agora temos um número k de variáveis preditoras.

11.1.3 Transformando dados para análises múltiplas

O método para criarmos modelos de regressão múltipla e a sua interpretação são praticamente os mesmos já estudados acima. Contudo, quando estamos pondo em jogo muitas variáveis de espectros diferentes, estaremos superestimando as que têm maior espectro. Para que você

entenda onde quero chegar, vamos dar um exemplo com três variáveis distintas: (1) idade de crianças no maternal, que normalmente varia de 1 a 3 anos; (2) dosagem de hemoglobina dessas crianças, que pode variar entre 11 a 15g/dL de sangue; (3) quantificação de células de defesa (leucócitos totais) circulantes no sangue, que varia razoavelmente entre 4.000 a 11.000 células/mm³ de sangue. Percebe o quão muda esses valores? As variações da variável "células de defesa" serão muito maiores, assim como os seus resíduos, seus desvios, a variância, seu desvio padrão e sua própria média. Tudo isso influencia para que essa variável, dentre as três citadas, acabe tendo um certo "destaque", o que deve ser evitado.

Para contornar isso, quando estivermos trabalhando com muitas variáveis, devemos usar mão de técnicas que coloquem elas no mesmo espectro. Existem duas mais comuns: a transformação para o escore Z ou a normalização dos dados. O escore Z tem uma certa ligação com a distribuição normal padrão, pois quando transformamos os dados para esse escore, a média tende a zero e o desvio padrão tende a um. Mas, é importante falar que empiricamente nem sempre os dados transformados se dispõem em uma distribuição normal. Contudo, essa transformação é uma boa tentativa para quem deseja trabalhar com estatística paramétrica, quando os dados brutos não estão na distribuição normal.

Para transformar os dados no escore Z, o R dispõe da função scale(). Mas podemos construir uma função nossa, nos baseando com a fórmula para a transformação de dados para escore Z abaixo,

$$z = \frac{x - \bar{x}}{s}$$

onde $\overline{x}$ é a média da variável e *s* é o desvio padrão amostral da variável. Vamos para o R construir assa função.

```
#construindo função para a transformação de dados
escoreZ <- function(x){
  return((x - mean(x)) / sd(x))
}
```

Vamos testá-la, observar o que retornará à partir dela e comparar com o que retorna a função scale(). Vamos criar um banco de dados sequenciado de 1 a 9, só a nível de ilustração.

```
valores <- seq(1,9,1)    #criando sequência de 1 a 9

#testando a função que criamos
escoreZ(valores)
[1] -1.4605935 -1.0954451 -0.7302967 -0.3651484  0.0000000  0.3651484  0.7302967
1.0954451  1.4605935
```

```
#Usando a função scale()
scale(valores)
 [1,] -1.4605935
 [2,] -1.0954451
 [3,] -0.7302967
 [4,] -0.3651484
 [5,]  0.0000000
 [6,]  0.3651484
 [7,]  0.7302967
 [8,]  1.0954451
 [9,]  1.4605935
```

Se você reparar, os valores transformados são exatamente os mesmos, só que são retornados pelo R de formas diferentes. Você pode usar qualquer uma das duas formas para transformar seus dados no escore Z. Mas, vamos continuar nosso aprendizado e observar a média e o desvio padrão dos dados transformados.

```
mean(escoreZ(valores))
[1] 0
sd(escoreZ(valores))
[1] 1

mean(scale(valores))
[1] 0
sd(scale(valores))
[1] 1
```

Tudo dando certo! Já esperávamos que a média fosse zero e o desvio padrão fosse um.

Outra forma de transformação dos dados é a normalização deles. Não confundir normalização em transformar na distribuição normal, apesar que isso possa acontecer. O objetivo fundamental da normalização é deixar os dados num espectro entre 0 e 1. Para isso, precisamos usar os valores mínimos e máximos, como mostra a fórmula abaixo.

$$x_{norm} = \frac{x - mínimo}{máximo - mínimo}$$

No R *base* não uma função própria para normalizar os dados, mas podemos construir a nossa própria função facilmente. Ficará assim:

```
normalizar <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

normalizar(valores)
[1] 0.000 0.125 0.250 0.375 0.500 0.625 0.750 0.875 1.000
```

Perceba que os nossos dados de 1 a 9 se transformaram de 0 a 1.

11.1.4 Criando modelos de regressão linear múltipla e interpretando seus resultados

Para a gente exemplificar a criação e interpretação de modelos de regressão linear múltipla, recorreremos ao banco chamado: swiss. Ele é um conjunto de dados incorporado ao R e contém informações sobre indicadores de fertilidade e socioeconômicos em estados da Suíça em 1888. As variáveis e seus significados contidos no banco são: (1) "Fertility": taxa de fertilidade, que representa o número médio de filhos nascidos vivos por mulher em idade fértil (15-49 anos) em cada cantão (tipo de divisão territorial) da Suíça em 1888; (2) "Agriculture": porcentagem de habitantes envolvidos em atividades agrícolas como ocupação principal em cada estado; (3) "Examination": porcentagem de recrutas masculinos que passaram em um exame médico. Esse exame era realizado para determinar a aptidão física para o serviço militar obrigatório na Suíça naquela época; (4) "Education": porcentagem de habitantes com educação acima do nível primário (educação secundária e superior combinadas); (5) "Catholic": porcentagem de habitantes que eram católicos; (6) "Infant.Mortality": taxa de mortalidade infantil, que representa o número de mortes de crianças menores de um ano por 1.000 nascidos vivos. Ou seja, um banco de dados típicos de estudos epidemiológicos.

Como ele já está incorporado ao R, abriremos ele usando a função data(), usaremos head() para vermos de forma geral o banco, transformaremos seus dados numéricos no escore Z usando a função scale(), transformaremos o conjunto de dados modificados em um *data frame* usando as.data.frame() e usaremos head() novamente para checarmos como os dados ficaram modificados. Se não usarmos a função as.data.frame(), o nosso banco modificado com scale() será do tipo matriz, o que não é interessante nesse momento.

```
data("swiss") #abrindo banco

head(swiss[,1:3])
             Fertility Agriculture Examination
Courtelary        80.2        17.0          15
Delemont          83.1        45.1           6
Franches-Mnt      92.5        39.7           5
Moutier           85.8        36.5          12
Neuveville        76.9        43.5          17
Porrentruy        76.1        35.3           9


#transformando dados
swiss_modif <- as.data.frame(scale(swiss))

head(swiss_modif[,1:3])
             Fertility Agriculture Examination
```

```
Courtelary   0.8051305  -1.4820682 -0.18668632
Delemont     1.0372847  -0.2447942 -1.31480509
Franches-Mnt 1.7897846  -0.4825622 -1.44015162
Moutier      1.2534283  -0.6234617 -0.56272591
Neuveville   0.5409551  -0.3152440  0.06400674
Porrentruy   0.4769125  -0.6762990 -0.93876550
```

Na Figura 11.2 encontraremos à esquerda os histogramas das variáveis do banco swiss com suas escalas originais. É possível observar que os espectros mudam bastante, deixando os seus valores obviamente mais espalhados graficamente. Quando alteramos a escala usando o escore Z (swiss_modif), percebemos à direta da figura que os histogramas se sobrepõem, porque as escalas agora estão iguais e centradas no valor zero. Não teremos assim o efeito da amplitude de média e desvio padrão interferindo no cálculo dos betas da regressão. Em resumo: agora estão todas as variáveis em "igual" condição.

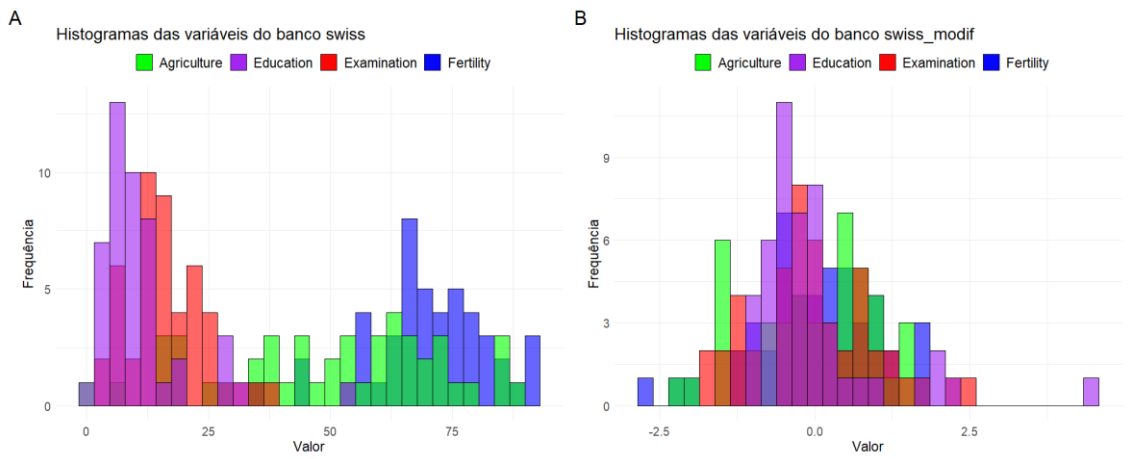


Figura 11.2. Histogramas demonstrando a diferença de espectros entre variáveis originais e modificadas pelo escore Z.
A - dados originais; B - dados modificados.

Agora, para criar nosso modelo usaremos ainda a função lm(). Na fórmula dentro da função, utilizaremos o ponto (.) após o til para indicar que queremos pôr na criação do nosso modelo todas as variáveis, exceto a que escolhemos como dependente (que deve ficar sempre à esquerda do til). Mais à frente trabalharemos no filtro de variáveis independentes, não se preocupe.

Na criação do nosso modelo, a variável relacionada à educação será o nosso desfecho e avaliaremos a associação ele com as demais (independentes). Lembrando: quanto maior o valor em Education, maior a porcentagem de pessoas com nível primário naquela região.

```
#criando o modelo de regressão linear múltipla
modelo_geral <- lm(Education ~ ., data = swiss_modif)
```

```
#analisando resultados
summary(modelo_geral)
Call:
lm(formula = Education ~ ., data = swiss_modif)

Residuals:
    Min      1Q  Median      3Q     Max
-1.1851 -0.2466 -0.0297  0.2923  1.1750

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       2.414e-16  7.444e-02   0.000 1.000000
Fertility        -5.307e-01  1.115e-01  -4.758 2.43e-05 ***
Agriculture      -3.836e-01  1.060e-01  -3.619 0.000804 ***
Examination       3.483e-01  1.356e-01   2.569 0.013922 *
Catholic          4.347e-01  9.323e-02   4.663 3.29e-05 ***
Infant.Mortality  6.182e-02  8.600e-02   0.719 0.476305
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5104 on 41 degrees of freedom
Multiple R-squared:  0.7678,    Adjusted R-squared:  0.7395
F-statistic: 27.12 on 5 and 41 DF,  p-value: 5.223e-12
```

Segundo os resultados, quanto maior o nível educacional, menor é o nível da taxa de fertilidade entre os cantões, pois o beta para “Fertility” foi negativo e obteve p-valor significativo. Podemos falar o mesmo para a porcentagem de habitantes na zona rural, ou seja, quanto maior o nível educacional, menor a proporção de pessoas em ocupações agrícolas tinha aquele cantão. Pelo que os resultados mostram, na Suíça, em 1888, as pessoas com maior nível escolar estavam mais presentes na zona urbana.

Houve associação positiva entre os níveis de educação e as proporções do exame de recrutamento militar e proporções de católicos. Ou seja, quanto maiores as taxas educacionais, maior era o exame e maior era a proporção de católicos na Suíça, naquela época. Contudo, perceba que a variável “Catholic” está mais associada ao desfecho do que “Examination”, uma vez que seu p-valor foi menor.

Já em relação à taxa de mortalidade infantil, não houve qualquer associação aparente, segundo o nosso modelo de regressão linear múltipla. Seu p-valor não foi significativo e isso indica que estatisticamente o beta relacionado a essa variável é igual a zero, não alterando expressivamente a variável desfecho.

Analisando o R^2, agora de preferência o ajustado, podemos dizer que as variáveis em jogo respondem a 73,95% da variação das taxas educacionais na Suíça (em 1888).

11.1.5 Comparando modelos de regressão

Como você tenha talvez intuído, as variáveis que são selecionadas para compor o modelo de regressão influenciam no produto final. Dessa forma, nada mais interessante do que comparar modelos de regressão que tenham o mesmo desfecho para que ao final possamos escolher talvez o mais adequado à nossa realidade.

Para compararmos modelos preditivos, é importante que se diga que devemos comparar regressões construídas do mesmo banco de dados. Imaginemos a situação de construirmos dois modelos, um usando banco de dados vindos da área da saúde e um outro modelo que utilizou-se um banco vindo da indústria (variáveis vindas dos parâmetros de construção de vergalhões para a engenharia civil). São variáveis bem diferentes, com espectros bem diferentes, com interpretações totalmente distintas. Não faz qualquer sentido a comparação.

Por conta disso, na prática, vamos construir dois modelos que se baseiam no banco de dados "swiss" que usamos acima. Contudo, diferente do que já realizamos, não vamos adicionar todas as variáveis independentes que utilizamos anteriormente. Para isso, selecionaremos cada variável desejada e colocaremos o sinal de adição (+) entre elas. É, a ideia é essa, adicionando variáveis independentes. Ao final de cada modelo, vamos usar a função summary() para verificarmos os resultados.

```
modelo1 <- lm(Education ~ Fertility + Agriculture + Catholic + Infant.Mortality, data
= swiss_modif)

summary(modelo1)
Call:
lm(formula = Education ~ Fertility + Agriculture + Catholic +
    Infant.Mortality, data = swiss_modif)

Residuals:
     Min       1Q   Median       3Q      Max
-1.08190 -0.28915 -0.07874  0.25931  1.33734

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       3.350e-16  7.925e-02   0.000 1.000000
Fertility        -6.765e-01  1.022e-01  -6.617 5.14e-08 ***
Agriculture      -5.404e-01  9.226e-02  -5.857 6.37e-07 ***
Catholic          3.614e-01  9.450e-02   3.825 0.000428 ***
Infant.Mortality  8.614e-02  9.100e-02   0.947 0.349243
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5433 on 42 degrees of freedom
Multiple R-squared:  0.7305,    Adjusted R-squared:  0.7048
F-statistic: 28.46 on 4 and 42 DF,  p-value: 1.804e-11
```

Observe que ao criamos o primeiro modelo, utilizamos ainda a variável "Education" como dependente, e "Fertility", "Agriculture", "Catholic", "Infant.Mortality" como independentes. Observe também que o banco que usamos foi o banco modificado para o escore Z.

```
modelo2 <- lm(Education ~ Fertility + Agriculture + Catholic, data = swiss_modif)

summary(modelo2)
Call:
lm(formula = Education ~ Fertility + Agriculture + Catholic,
    data = swiss_modif)

Residuals:
     Min       1Q   Median       3Q      Max
-1.04886 -0.30701 -0.06946  0.33819  1.34894

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.738e-16  7.916e-02   0.000  1.00000
Fertility   -6.351e-01  9.228e-02  -6.881 1.91e-08 ***
Agriculture -5.621e-01  8.926e-02  -6.298 1.35e-07 ***
Catholic     3.661e-01  9.426e-02   3.884  0.00035 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5427 on 43 degrees of freedom
Multiple R-squared:  0.7247,    Adjusted R-squared:  0.7055
F-statistic: 37.73 on 3 and 43 DF,  p-value: 4.123e-12
```

Para o segundo modelo, retiramos a variável independente "Infant.Mortality", que não se mostrou significativa no modelo 1.

Para modelos de regressões lineares, há pelo menos cinco métricas que podem ser utilizadas para compararmos modelos criados: (1) o coeficiente de determinação (R^2); (2) a raiz quadrática média; (3) o erro absoluto médio; (4) o critério de informação de Akaike (AIC); (5) o critério de informação bayesiano (BIC).

Já sabemos que o coeficiente de determinação é uma medida estatística que avalia a qualidade de ajuste de um modelo de regressão aos dados observados. Quanto mais próximo de 1 for o R^2, melhor é o ajuste do modelo. Enquanto que quanto mais próximo de zero, pior é o ajuste do modelo aos dados. Dessa forma, na comparação de dois ou mais modelos de regressão, o modelo com um R^2 mais elevado é geralmente considerado como sendo o melhor ajuste aos dados observados.

No entanto, é importante ter em mente que o R^2 sozinho não é uma medida completa da qualidade do modelo e não leva em consideração outros fatores importantes, como a validade das

premissas do modelo, a robustez dos resultados, a interpretação dos betas e a adequação do modelo para fins de previsão. Por isso, vamos considerar as outras métricas.

Existem duas métricas que levam em consideração o erro, ou desvios, cometidos pelo modelo de predição criado: (1) a raiz quadrática média (RMSE, do inglês *Root Mean Squared Error*), e (2) o erro absoluto médio (MAE, do inglês *Mean Absolut Error*). As fórmulas de como calculá-los de um e de outro estão logo abaixo.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

A chave para interpretar RMSE e MAE é que suas fórmulas se baseiam no desvio/erro de cada modelo. Essas métricas têm intervalos de zero a infinito, sendo que quanto maior o seu valor, maior também será o erro respectivo. Assim, quando estivermos comparando modelos, certamente será melhor o modelo que tem menor erro, então, quanto menor for RMSE ou MAE, melhor será o modelo.

O critério de informação de Akaike (AIC), por sua vez, é uma medida de ajuste que penaliza modelos mais complexos, adicionando uma penalidade proporcional ao número de parâmetros (betas) estimados. Interpretativamente falando, quanto menor o valor do AIC, melhor o modelo é em termos de ajuste aos dados. Já o critério de informação bayesiano (BIC), por outro lado, é uma variante do AIC que inclui uma penalidade mais rigorosa para modelos com mais parâmetros. Ele é considerado ainda mais conservador do que o AIC em termos de penalização para a complexidade do modelo. Para interpretarmos, assim como o AIC, quanto menor o valor do BIC, melhor o modelo é considerado em termos de ajuste aos dados.

A tabela 11.1 resume a interpretação das métricas quando estamos comparando modelos:

Tabela 11.1 Resumo interpretativo de métricas na comparação de modelos.

Métrica	Interpretação
R^2	Quanto mais próximo de 1, melhor.
RMSE	Quanto menor, melhor
MAE	Quanto menor, melhor
AIC	Quanto menor, melhor
BIC	Quanto menor, melhor

Na prática, o R^2 já faz parte dos resultados diretos da função summary(). Para acharmos o AIC e BIC, usaremos as funções AIC() e BIC() do R base. Já RMSE e MAE, utilizaremos as funções rmse() e mae() do pacote Metrics. Caso você não o tenha, deve lembrar-se de instalá-lo previamente. Vamos comparar os três modelos gerados anteriormente: modelo_geral, modelo1 e modelo2. Assim, poderemos conhecer qual dentre as três é a melhor combinação de variáveis para explicar a taxa educacional da Suíça.

Para gerarmos as métricas de RMSE e MAE precisamos de um banco que chamamos comumente de banco teste, ou seja, uma subdivisão do banco original, cujos dados não foram usados para criar o modelo. Vamos supor a seguinte situação: tenho um total de 500 pacientes em um banco de dados; seleciono 50 entre eles para gerar o banco teste e os outros 450 será o meu banco de treino para gerar o modelo de regressão. Como eu preciso subdividir o banco de dados original, dependendo da finalidade e da quantidade amostral limitada que se tem disponível, a comparação entre modelos poderia se basear apenas no R^2, AIC e BIC.

Para fins demonstrativos, iremos criar um banco de teste selecionando aleatoriamente algumas observações do banco original. Essa é uma das formas de seleção para compor o banco teste, mas devo salientar que talvez não seja a melhor das possibilidades de seleção, pois posso acabar selecionando mais indivíduos de uma determinada condição do que de outra e assim eu estaria enviesando as métricas. O ideal é selecionar aleatoriamente indivíduos de cada condição (classe) da variável desfecho.

Na prática, além de instalar e abrirmos a biblioteca Metrics, faremos a seleção aleatória de quinze números inteiros usando sample(), que representarão a nossa amostragem para teste. Criaremos então o banco teste chamado df_teste.

```
install.packages('Metrics')
library(Metrics)

n_teste <- sample(15)

#linhas selecionadas
n_teste
 [1] 12 10  6  5  9 15 13  8  4  2 11 14  3  7  1


#banco teste
df_teste <- swiss_modif[n_teste,]
```

Usando sample(), é muito provável que as linhas selecionadas por você tenham sido outras, mas na maioria das vezes isso não acarreta em grandes mudanças nos resultados finais, principalmente quando o banco de dados tem disponível um n amostral grande.

Vamos agora criar um *data frame* inicialmente vazio chamado "result", para que possamos incorporar nele os resultados de cada métrica. Assim ficará mais fácil e didático para interpretarmos os resultados. Então, vamos aos comandos para gerar nossos resultados finais de comparação entre modelos:

```
result <- data.frame(row.names = c('R2','AIC','BIC','RMSE','MAE'),
  modelo_geral = rep(NA,5),
  modelo1 = rep(NA,5),
  modelo2 = rep(NA,5)
)

result[1,1] <- summary(modelo_geral)$r.squared
result[1,2] <- summary(modelo1)$r.squared
result[1,3] <- summary(modelo2)$r.squared

result[2,1] <- AIC(modelo_geral)
result[2,2] <- AIC(modelo1)
result[2,3] <- AIC(modelo2)

result[3,1] <- BIC(modelo_geral)
result[3,2] <- BIC(modelo1)
result[3,3] <- BIC(modelo2)

result[4,1] <- rmse(df_teste$Education, predict(modelo_geral,df_teste))
result[4,2] <- rmse(df_teste$Education, predict(modelo1,df_teste))
result[4,3] <- rmse(df_teste$Education, predict(modelo2,df_teste))

result[5,1] <- mae(df_teste$Education, predict(modelo_geral,df_teste))
result[5,2] <- mae(df_teste$Education, predict(modelo1,df_teste))
result[5,3] <- mae(df_teste$Education, predict(modelo2,df_teste))
```

Perceba que as funções rmse() e mae() precisão da predição, que para isso usamos a função predict() incorporada ao código. Lembrando que para gerar predição, precisamos do modelo e de dados novos para isso (df_teste). Perceba ainda que tanto rmse(), quanto mae(), precisam da variável dependente presente no banco de teste (df_teste$Education).

Vamos usar a função round() com o argumento 2 (que está vinculado ao número de dígitos), para arredondar os números presentes nos resultados para dois dígitos após a vírgula. E, teremos o seguinte:

```
round(result,2)
     modelo_geral modelo1 modelo2
R2           0.77    0.73    0.72
AIC         77.73   82.75   81.74
BIC         90.68   93.85   90.99
RMSE         0.40    0.46    0.46
MAE          0.33    0.35    0.36
```

Recorra à Tabela 11.1 para interpretação. De forma sucinta, o modelo geral tem R^2 maior, então ele explica melhor as variações das taxas educacionais. Além disso, as demais métricas têm valores mais baixos, demonstrando que nesse modelo há desvios/erros menores que modelo1 e modelo2.

11.1.6 Covariáveis

As covariáveis em modelos de regressão são variáveis independentes adicionais. Elas são incluídas em um modelo estatístico para controlar o efeito de possíveis fatores de confusão ou variáveis de interferência. Elas são utilizadas para explicar parte da variação observada na variável dependente que não é explicada pelas variáveis independentes principais do modelo. Então, é desejável e coerente que o coeficiente de determinação aumente quando as usamos.

Por exemplo, em um estudo que investiga o efeito de um medicamento sobre a pressão arterial, a idade dos participantes pode ser uma covariável para controlar o efeito da idade na pressão arterial, isolando assim o efeito específico do medicamento. Outros exemplos de covariáveis podem ser sexo, nível socioeconômico, histórico médico, ou qualquer outra variável que seja considerada relevante para controlar possíveis efeitos de confusão em uma análise estatística.

Na prática, as covariáveis entrarão na fórmula de criação do modelo como uma variável independente comum. Mas entenda, quando uma covariável entra no modelo, ela está apenas ajustando os betas.

Há variáveis que irão piorar o desempenho do modelo, são as variáveis de ruido, que comumente não têm qualquer relação com o desfecho. Devemos identifica-las e removê-las se preciso, dependendo da nossa realidade. Mas, existem as variáveis para melhorar o desempenho do modelo, são as covariáveis, que já sabemos que para o determinado desfecho ela é bem relacionada, já é de conhecimento da literatura que ela influencia na variável dependente. Então, podemos pôr elas somente para regular o modelo preditivo final.

Dando um exemplo genérico de como montar uma regressão com covariáveis no R, seria muito simples e básico:

```
lm(efeito_medicamento ~ pressao_arterial + idade + sexo + socioeconômico + histórico,
data = banco_de_dados)
```

A única diferença aqui é que o interesse do pesquisador é avaliar o efeito do medicamento segundo a pressão arterial. As demais variáveis estão como coadjuvantes.

Para que você entenda de uma vez por todas, vou dar um último exemplo. Vamos supor que um geneticista esteja estudando associações genéticas com um câncer específico. As variáveis genéticas que ele tem disponíveis são todas baseadas nos nucleotídeos: A (adenina), T (timina), G (guanina) e C (citosina), que serão regiões do DNA. Mas, esse pesquisador já sabe que esse determinado câncer é mais comum em homens do que em mulheres. Dessa forma, ele coloca a variável sexo na fórmula de regressão apenas para ajustar o modelo. Nesses resultados, não se espera que ele venha tirar grandes conclusões da relação "determinado câncer" vs. "sexo", porque isso já se sabe bastante.

11.1.7 multicolinearidade

Já que falamos em análises de regressão múltipla, precisamos minimamente conversar sobre o conceito de multicolinearidade. Isso ocorre quando duas ou mais variáveis independentes em um modelo de regressão múltipla estão altamente correlacionadas entre si. Isso pode causar problemas na interpretação dos resultados do modelo, uma vez que pode dificultar a identificação dos efeitos individuais dessas variáveis sobre a variável dependente.

As vezes pensamos que quanto mais usarmos variáveis mais correlacionadas com o desfecho, melhor será nosso modelo de regressão. Na verdade, se assim fosse, estaríamos forçando o modelo a superestimar a realidade. Portanto, devemos evitar a multicolinearidade.

Há algumas maneiras de identificar a presença de multicolinearidade em um modelo de regressão múltipla:

1. Avaliar a matriz de correlação: se houver correlação forte (positiva ou negativa) entre duas ou mais variáveis independentes, já é um indicativo de multicolinearidade.
2. Avaliar a variância da inflação do fator (VIF): a VIF é uma medida estatística que quantifica a quantidade de inflação na variância de um coeficiente de regressão devido à multicolinearidade. Valores de VIF acima de 5 sugere-se multicolinearidade; acima de 10 sugere sério problema de multicolinearidade. Quanto maior o valor do VIF, maior a presença de multicolinearidade.
3. Testes de significância dos coeficientes de regressão: a presença de multicolinearidade pode afetar a significância estatística dos coeficientes de regressão. Se a multicolinearidade for alta, os coeficientes de regressão podem ter erros-padrão elevados, resultando em valores t baixos e p-valores altos, o que pode levar à não significância estatística dos coeficientes.

4. Análise do R^2 ajustado: se a multicolinearidade estiver presente, é possível que o R^2 ajustado seja muito alto, o que pode indicar que o modelo está superajustado devido à duplicação de efeitos das variáveis independentes.
5. Análise dos sinais e magnitude dos coeficientes: a multicolinearidade pode afetar a interpretação dos coeficientes de regressão em termos de suas magnitudes e sinais. Coeficientes de regressão com magnitudes inconsistentes ou com sinais opostos ao esperado podem ser um indicativo de multicolinearidade.

É importante notar que a identificação da multicolinearidade não implica necessariamente na necessidade de remoção das variáveis correlacionadas do modelo. Há várias formas para lidar com esse problema, mas fogem um pouco do escopo deste livro. O mais importante é ter uma análise cuidadosa do contexto do estudo e dos objetivos de pesquisa para uma interpretação adequada dos resultados em presença de multicolinearidade. Ir testando combinações de variáveis a serem implantadas no modelo pode ser uma boa saída. Pesquise também por métodos de seleção de variáveis, como regressão Ridge e LASSO. Usar os escores dos resultados de Análise de Componentes Principais também pode ser uma solução; estamos falando aqui em redução de dimensionalidade. Se a sua pesquisa necessitar desse cuidado, busque primeiramente sobre esses assuntos.

Abaixo, vamos demonstrar um caso simples de multicolinearidade. É um banco de dados complementar ao que utilizamos mais acima, do rendimento de carcaça. Chamaremos de "reg_mult", você pode cria-lo ou baixar via Github da página do livro. A variável "temp_ani" indica a temperatura corpórea do animal, em °C; "temp_amb" é a temperatura ambiente no qual o suíno está inserido, em °C; "fluxo_vent" é o fluxo de vento que esses animais recebem como forma de bem-estar animal; e, "carin" é a quantidade de carinho que esse animal recebe espontaneamente de humanos ao longo do dia. Temos, portanto:

```
#criando banco
reg_mult <- data.frame(
  idade = c(6,7,8,7,10,9,12,11,14,15,6,11,14),
  carc = c(89,88,90,98,89,112,110,124,130,136,78,100,127),
  temp_ani = c(37.6,38.6,39.1,38.9,39.3,38.7,39.1,39.4,39.6,40.3,37.1,39.7,40.2),
  temp_amb = c(26,26.5,26.7,27,26.4,27.3,27.1,28,27.3,26.1,25.8,26.9,27.1),
  fluxo_vent = c(0.2,0.3,0.1,0.3,0.1,0.2,0.3,0.1,0.2,0.1,0.1,0.3,0.2),
  carin = c(7,6,8,5,3,7,2,2,1,0,6,2,1)
)

#ou importando o banco
library(rio)
reg_mult <- import('pasta_de_arquivos/reg_mult.txt')

#mudando as escalas para escore Z
```

```
reg_mult <- as.data.frame(scale(reg_mult))

#criando o modelo de regressão linear múltiplo
result_rlm <- lm(carc ~ ., data = reg_mult)
```

Quando analisamos os betas da regressão, observamos que o rendimento de carcaça se mostrou associado à idade, mas com um p-valor bem próximo do limítrofe para α = 5%. Uma associação baixa. Veja:

```
summary(result_rlm)

Call:
lm(formula = carc ~ ., data = df2)

Residuals:
     Min       1Q   Median       3Q      Max
-0.62737 -0.35867  0.01267  0.24504  0.62660

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.894e-16  1.405e-01   0.000   1.0000
idade        1.086e+00  4.531e-01   2.397   0.0477 *
temp_ani    -1.798e-01  3.102e-01  -0.580   0.5804
temp_amb     2.712e-01  1.709e-01   1.587   0.1566
fluxo_vent  -2.960e-02  1.577e-01  -0.188   0.8564
carin        1.669e-01  3.387e-01   0.493   0.6372
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5065 on 7 degrees of freedom
Multiple R-squared:  0.8503,    Adjusted R-squared:  0.7434
F-statistic: 7.954 on 5 and 7 DF,  p-value: 0.008362
```

Vamos rodar a matriz de correlação e analisar quais variáveis estão correlacionadas entre si sando a função cor(). Usaremos a round() com argumento 2 para reduzir as casas decimais.

```
round(cor(reg_mult),2)
           idade  carc temp_ani temp_amb fluxo_vent carin
idade       1.00  0.89     0.87     0.38      -0.09 -0.89
carc        0.89  1.00     0.76     0.54      -0.08 -0.75
temp_ani    0.87  0.76     1.00     0.47       0.02 -0.75
temp_amb    0.38  0.54     0.47     1.00       0.20 -0.30
fluxo_vent -0.09 -0.08     0.02     0.20       1.00 -0.01
carin      -0.89 -0.75    -0.75    -0.30      -0.01  1.00
```

Perceba que essa matriz é “espelhada”. Os coeficientes de correlação estão repetidos segundo a diagonal principal. Sempre a diagonal principal será comporta por 1, pois a correlação de uma variável com ela mesmo sempre terá uma correlação perfeita. Mas, perceba que idade e rendimento de carcaça têm correlação forte (r = 0,89). Idade e a temperatura corpórea do animal também tem correlação forte (r = 0,87). Idade e a quantidade de carinho recebido têm correlação

forte, porém negativa, ou seja, quanto mais jovem for o animal, mais carinho ele tende a receber (r = -0,89). Carcaça e temperatura corpórea também apresentou correlação forte. Carcaça e quantidade de carinho diário. Temperatura corpórea e carinho diário. Caramba, são tantas correlações significativas, por que isso não ficou evidente nos resultados de regressão?

Vamos calcular a VIF das variáveis e analisar se há multicolinearidade sugestiva para alguma delas. Vamos utilizar a função vif() do pacote car (do inglês *Companion to Applied Regression*) para isso.

```
install.packages('car')
library(car)

vif(result_rlm)
     idade   temp_ani   temp_amb fluxo_vent      carin
  9.602977   4.499443   1.366555   1.163089   5.364058
```

Pois bem, nosso problema está aqui. As variáveis idade e carinho apresentaram valores de VIF maiores que 5. É sugestivo a multicolinearidade. Fluxo de vento obteve menor VIF e observe que não há correlações fortes dessa variável com nenhuma outra.

11.2 REGRESSÃO LOGÍSTICA

Se você reparar bem, até agora trabalhamos as regressões utilizando sempre variáveis numéricas. Em regressões lineares, em especial, os desfechos são variáveis contínuas ou discretas. Mas, e se desejássemos usar como desfecho uma variável categórica? Bom, podemos codificar as classes dessa variável numericamente, contudo, existe uma forma conveniente de fazermos isso: dicotomizar. Vamos supor que temos a variável índice de massa corpórea (IMC), mas desejamos construir um modelo de regressão, colocando-a como variável dependente quem está com sobrepeso e quem não está. Podemos codificar a variável usando o ponto de corte de 25 kg/m². Quem tiver valor abaixo de 25 kg/m², codificaremos como 0, e quem tiver 25 kg/m² ou mais, codificaremos como 1. Acabamos de fazer a dicotomia 0 e 1. As dicotomizações ajudam muito na interpretação de diversos problemas.

Isso é mais simples quando temos a presença de uma doença ou não. Se a pessoa, animal ou planta estiver sadia, codificamos com 0. Se estiver doente, codificaremos com 1. Não é bem uma regra, mas há uma certa convenção de codificarmos sempre a condição desfavorável com o valor mais elevado. Também ajudará na hora da interpretação de resultados. Isso vale para variáveis dependentes como independentes. Essa transformação da variável para 0 e 1 você também pode achar como *dummy*.

Para analisarmos variáveis desse universo simplificadamente dicotomizado, é mais interessante usarmos um algoritmo de regressão que nos retorne uma predição mais específica: 0 ou 1. É assim que a regressão logística trabalha. Ela é amplamente utilizada na área da saúde porque os desfechos comumente são doenças (estar ou não com aquela condição), portanto, uma dicotomizações.

Os códigos e a forma interpretativa da regressão logística são muito parecidos com a regressão linear. Apenas modificaremos a função que gera o modelo e alguns parâmetros e acrescentaremos a ideia de *odds ratio* (ou razão de chance) e seu intervalo de confiança para mensurar o nível de associação quando houver.

Para exemplificar, vamos imaginar um banco de dados coletados por pesquisadores que estavam trabalhando com pacientes diabéticos do tipo I. O banco se chamará "dadosDM" e teremos três variáveis independentes: dosagem de glicose sérica de jejum (0 = até o valor de referência; 1 = glicose elevada), dosagem de insulina (0 = valores normais; 1 = valores considerados baixos) e tgp (TGP ou ALT = alanina aminotransferase; dosagem em mg/dL), um marcador bioquímico muito conhecido da função hepática. O desfecho "DM" (diabetes melitos), está codificado com 0 os indivíduos saudáveis para essa condição e 1 para os pacientes com diabetes melitos do tipo I.

```
#importando dados
library(rio)
dadosDM <- import('pasta_de_arquivos/dadosDM.txt')


head(dadosDM)
  DM gli insu tgp
1  1   1    0  46
2  1   1    0  28
3  1   1    0  20
4  1   1    0  33
5  1   1    0   8
6  1   1    0  33
```

Como visto, usando a função head(), as variáveis não estão no mesmo espectro. Vamos reescalar os dados com a função scale(), exceto a variável dependente de interesse, que é DM.

```
#modificado as escalas das variáveis independentes
dadosDM[-1] <- as.data.frame(scale(dadosDM[-1]))

head(dadosDM)
  DM      gli      insu          tgp
1  1 1.082931 -1.061382  1.335475648
2  1 1.082931 -1.061382 -0.004914765
3  1 1.082931 -1.061382 -0.600643837
4  1 1.082931 -1.061382  0.367415905
```

```
5  1 1.082931 -1.061382 -1.494237446
6  1 1.082931 -1.061382  0.367415905
```

Vamos prosseguir agora criando nosso modelo de regressão logística utilizando a função glm(), que é usada para modelos de regressão generalizados. Para a função identificar que queremos uma regressão logística, devemos utilizar no parâmetro "family" a seguinte descrição: binomial(link = "logit"). Vamos sumarizar os resultados.

```
#modelo de regressão logística
result_rlog <- glm(DM ~ ., data = dadosDM, family = binomial(link = "logit"))

summary(result_rlog)
Call:
glm(formula = DM ~ ., family = binomial(link = "logit"), data = df3)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1576  -0.5795  -0.0582   0.4622   1.9484

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.09854    0.09352   1.054 0.292002
gli          1.45348    0.14345  10.132  < 2e-16 ***
insu        -0.49147    0.14169  -3.469 0.000523 ***
tgp          0.01884    0.09230   0.204 0.838219
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1386.29  on 999  degrees of freedom
Residual deviance:  792.16  on 996  degrees of freedom
AIC: 800.16

Number of Fisher Scoring iterations: 4
```

Na fórmula da função, perceba que poderíamos ter usado: DM ~ gli + insu + tgp. Daria no mesmo resultado.

Como podemos observar nos resultados gerados pela regressão logística, ter valores elevados de glicose está altamente associado positivamente com a DM (olhe o valor de beta positivo), ou seja, quanto maior a taxa de glicose, maior a probabilidade de ser diabético do tipo I. Seu p-valor associados é muito menor do que 0,001 (especificamente, < 2e-16). Já a taxa de insulina está associada negativamente (olhe o valor de beta), ou seja, quanto menor a taxa de insulina, maior a probabilidade de ser diabético do tipo I (p-valor = 0.000523). Já a dosagem de TGP não está associada à DM.

Na regressão logística, ainda há uma métrica que nos ajuda na interpretação desses resultados, que é a razão de chance (ou *odd ratio* - OR) e o seu intervalo de confiança. Para que compreendermos do que se trata, vamos entender primeiro o que é chance. Vamos supor que amanhã a probabilidade de chover seja de 70%. Então, a chance de chover amanhã é 70/100, ou seja, 70 em cada 100 possibilidades.

A razão de chance em si é a divisão da chance de ter um efeito sobre a chance de não ter o efeito.

$$razão\ de\ chance = \frac{chance\ de\ ter\ efeito}{chance\ de\ não\ ter\ efeito}$$

Dessa forma, se a razão de chance for igual a 1, a chance de ter o efeito e de não ter o efeito é exatamente a mesma, concorda? Um valor dividido por ele mesmo é igual a 1. Se a razão de chance for maior do que 1, então a chance de ter o efeito é maior do que não ter o efeito. Já, se a razão de chance for menor do que 1, então a chance de não ter o efeito é maior do que ter o efeito. Aqui, não há razão de chance com valores negativos. Seu espectro varia de maior que 0 a infinito. Quando acima informei que os desfechos desfavoráveis são normalmente codificados com o valor maior, é porque interfere principalmente nessa análise.

Como não há um p-valor relacionado à razão de chance como teste de hipótese para afirmarmos estatisticamente que ele é igual a zero ou não, usamos o seu intervalo de confiança (IC) para decidirmos isso. Se no intervalo o valor 1 ocorrer, então consideramos estatisticamente a razão de chance igual a 1, ou ainda, podemos dizer que provavelmente o efeito é nulo.

Vamos retomar nosso exemplo com o banco sobre diabetes, vamos calcular as razões de chance e seus intervalos de confiança usando a função exp() e interpretar cada resultado. A função cbind() está presente para juntar os valores demonstrativamente. Já as funções coef() e confint() retornaram os coeficientes e seus intervalos, respectivamente.

```
exp(cbind(OR=coef(result_rlog), confint(result_rlog)))
                    OR     2.5 %    97.5 %
(Intercept) 1.1035628 0.9197530 1.3279678
gli         4.2779930 3.2548675 5.7258595
insu        0.6117249 0.4665420 0.8152664
tgp         1.0190233 0.8503178 1.2216351
```

Vamos a aprender a interpretar com os nossos próprios resultados. Para glicose, temos uma razão de chance de 4,28 (3,25-5,73); percebeu que no IC a razão de chance está contida? Dessa forma, podemos dizer que para quem tem glicose elevada a chance de ser diabético do tipo I é 4 vezes maior do que quem não tem glicose elevada no sangue; podemos falar também que a chance

é 428% maior. Perceba também que o IC o valor 1 não está presente, então, existe realmente efeito. Há analistas que diriam que ter glicose elevada é um fator de risco para a diabetes, o que é compreensível biologicamente falando, mas é importante falar que risco de fato é uma métrica epidemiológica, a qual será abordada em capítulos mais à frente.

Em relação a insulina, o efeito é inverso, pois o valor de OR foi abaixo de 1 (e isso confirma com o valor do beta da equação). Podemos dizer que quem tem baixas taxas de insulina no sangue há uma chance de 61,17% de ter diabetes do tipo I. Da mesma forma, podemos também dizer que ter taxas elevadas de insulina no sangue é um fator protetor. Em outras palavras, os indivíduos com valores mais elevados de insulina apresentariam um risco menor de desenvolver diabetes do tipo I em comparação com aqueles com valores mais baixos de insulina sérica.

Já a dosagem de TGP não influencia no desfecho. Já sabemos que o p-valor do beta é estatisticamente igual a zero. A OR e seu intervalo de confiança deu, respectivamente: 1,02 (0,85-1,22). Consegue observar que o valor 1 está contido no intervalo? Dessa forma, não há efeito da dosagem de TGP na diabetes melitos do tipo I, segundo a nossa amostragem.

Para que esses resultados fiquem todos mais atrativos para analisar, vamos juntá-los usando a função cbind(). Usaremos round() para colocar os valores com apenas duas casas decimais.

```
round(cbind(summary(result_rlog)$coefficients,
        exp(cbind(OR=coef(result_rlog),
                  confint(result_rlog)))),2)

            Estimate Std. Error z value Pr(>|z|)   OR 2.5 % 97.5 %
(Intercept)     0.10       0.09    1.05     0.29 1.10  0.92   1.33
gli             1.45       0.14   10.13     0.00 4.28  3.25   5.73
insu           -0.49       0.14   -3.47     0.00 0.61  0.47   0.82
tgp             0.02       0.09    0.20     0.84 1.02  0.85   1.22
```

Geralmente omitimos a linha “(Intercept)” e as colunas “std. Error” e “z value” quando mostramos nossos resultados ao público. O que mais nos importa é discutimos os valores dos betas (Estimate), p-valor dos betas (Pr(>|z|)), OR e seu intervalo de confiança com seus limites inferiores e superiores (respectivamente, 2.5% e 97.5%).

Capítulo 12

Teste de Associação Entre Variáveis Categóricas

Como vimos no capítulo anterior sobre regressão logística, podemos inferir associações entre o desfecho categorizado e variáveis independentes. Ocorre que as regressões não são os únicos métodos estatísticos que utilizamos para inferir associações.

Utilizamos teste de qui-quadrado e exato de Fisher para testar a hipótese de que há associação entre variáveis categóricas. Nesses testes estamos comparando proporções entre classes.

Vamos trabalhar aqui pelo menos de três testes diferentes, como o qui-quadrado de aderência, qui-quadrado de independência e o exato de Fisher, verificando a suas respectivas utilizações.

12.1 TESTE QUI-QUADRADO DE ADERÊNCIA

O senhor Gregor Johann Mendel, um frade agostiniano, nos honrou gloriosamente com suas pesquisas com plantas. Mendel, mesmo não conhecendo o que eram os ácidos nucleicos, ele conseguiu inferir que certas características poderiam ser herdadas, e havia uma frequência esperada de características diferentes na próxima geração.

Lembrando, ele trabalhou muito com ervilhas. Mendel reparou que se uma planta heterozigota (semente lisa) fosse cruzada com outra heterozigota, havia a probabilidade de 25% de surgir na próxima geração plantas que produziam ervilhas de sementes lisas (homozigotas), 50% de surgir plantas heterozigotas (também lisas), e 25% de surgirem plantas homozigotas recessivas (sementes enrugadas).

Na época, Mendel testou empiricamente diversas gerações até chegar em suas conclusões. Mas, hoje, graças ao matemático e também biólogo britânico Karl Pearson (o mesmo da correlação), nos agraciou com o teste de aderência. Nele, conseguimos estatisticamente inferir se os dados que foram observados aderem ao que se é esperado de uma teoria que envolva proporções.

Vamos exemplificar: quando Mendel verificou as proporções das ervilhas, certamente na prática ele não encontrou as proporções 25%, 50% e 25% redondo dessa forma. Ele pode ter encontrado proporções do tipo: 24,4%, 52,0% e 23,6%. A pergunta que fica, é: será que as proporções observadas na prática se equivalem estatisticamente à esperada? Pois se sim, a teoria de Mendel é válida.

A fórmula criada por Pearson para qui-quadrado (χ^2) é:

$$\chi^2 = \sum_{i=1}^{r} \frac{(O_i - E_i)^2}{E_i}$$

Os valores de O_i são as frequências observadas, enquanto que E_i são as frequências esperadas (no caso, teóricas). A letra *r* são as categorias presentes na variável de interesse. Caso você queira se aprofundar no assunto, você encontrará artigos expondo esse teste como *Chi-Square Goodness of Fit Test*.

Na prática, podemos utilizar a função chisq.test() para realizar todos os testes de qui-quadrado. Vamos pôr as proporções do exemplo de Mendel acima na forma decimal. Aqui, o teste de hipótese é:

> H_0: as proporções do que foi observado e do que foi esperado são estatisticamente iguais.
> H_1: as proporções são diferentes.

Então, se o p-valor for maior do que 0,05 (usando o nível de significância de 5%), aceitamos H_0 e concluímos que Mendel estava certo. Vamos verificar isso em um levantamento de dados de um aluno de biologia. Ele coletou e genotipou (analisar a constituição genética) de 48 amostras de uma determinada planta. Segunda a sua pergunta inicial, ele acha que esse marcador genético segue a lei da segregação independente que Mendel estabeleceu, portanto, os genótipos AA deve ter 25% de frequência, AC deve ter 50% e CC deve ter 25%. Sendo assim, vamos para o R.

```
#construindo um banco hipotético
observado <- c(rep('AA',12),rep('AC',26),rep('CC',10))
tabela_ob <- table(observado)

#as frequências absolutas encontradas de cada classe (genótipo)
tabela_ob
observado
AA AC CC
12 26 10

#a proporção esperada de cada genótipo
esperado <- c(0.25,0.50,0.25)

#teste qui-quadrado de aderência
chisq.test(tabela_ob, p = esperado)

      Chi-squared test for given probabilities

data:  tabela_ob

X-squared = 0.5, df = 2, p-value = 0.7788
```

O resultado traz um valor de χ^2 calculado relativamente baixo (0,5) e isso é um indício de aceite de H_0, mas a conclusão final vem do valor P. Como o p-valor foi maior do que 0,05, o resultado achado pelo aluno foi o mesmo achado por Mendel, pois as proporções entre o que foi observado e o que foi esperado são estatisticamente iguais, seguindo, portanto, a lei de Mendel. Perceba que a entrada de dados é a frequência bruta. A própria função se encarrega de todos os cálculos subsequentes necessários. Mas para que não fique dúvidas, a função calculará as frequências esperadas, e a partir disso, calculará o qui-quadrado pela fórmula exposta acima.

É preciso salientar aqui que o teste de qui-quadrado tem algumas limitações que devem ser consideradas ao utilizá-lo:

- Tamanho da amostra: o teste do qui-quadrado é mais confiável quando a amostra é grande o suficiente (superior, pelo menos, a 20 observações) para que as frequências observadas em cada categoria sejam pelo menos 5. Caso contrário, o teste pode não ser confiável.
- Dependência entre categorias: o teste do qui-quadrado de aderência assume que as categorias são independentes entre si. Se as categorias são dependentes, o teste pode produzir resultados imprecisos.

Nosso exemplo acima foi bem hipotético. Existem outras formas para trabalhar as frequências genéticas. Mas, se o seu interesse for responder se uma determinada amostragem adere a uma determinada proporção, estamos vendo aqui o teste correto. Na função você pode pôr as proporções no formato de fração ao invés de decimal, por exemplo: `c(1/4, 1/2, 1/4)`.

12.2 QUI-QUADRADO DE INDEPENDÊNCIA

Para entendermos a aplicação do teste de qui-quadrado de independência, vamos discutir um pouco sobre a tabela de contingência. Ela é feita quando estamos tratando de duas variáveis categóricas. Nessa tabela, aparecerá as frequências das relações entre as classes de cada variável.

Uma variável categórica minimamente analisável deve conter pelo menos duas classes, gerando minimamente uma dicotomia, portanto. Quando analisamos duas variáveis categóricas dicotômicas, estamos diante de uma tabela muito utilizada na bioestatística: a tabela 2x2.

A dicotomização facilita algumas interpretações, mas há a desvantagem da perda de informação. Se dividirmos em grupos pessoas quanto a cor da pele entre brancos e não brancos, perdemos a informação da diversidade étnica que está atrelada a esse quesito: pardo, negro, asiático, ameríndio, etc. Ou seja, tudo tem prós e contras.

Vamos supor que uma psicóloga esteja estudando dois métodos de aprendizagem: um método mais teórico e um método mais voltado para a prática. Ela deseja saber se há associação com o tipo de personalidade dos alunos: introvertido e extrovertido.

A hipótese que ela deve testar é:

> H_0: não há associação entre os métodos de aprendizagem e o perfil de personalidade dos alunos.
>
> H_1: há associação entre os métodos de aprendizagem e o perfil de personalidade dos alunos.

Vamos criar esse banco de dados da seguinte maneira: para métodos vamos reduzir os termos "teórico" para "Teor" e "prático" para "Prat". Vamos pôr a personalidade do tipo introvertido como "Introv" e extrovertido como "Extrov".

```
psi_df <- data.frame(
  metodo = c(rep('prat',10),rep('teor',60),rep('prat',30)),
  personalidade = c(rep('Introv', 50), rep('Extrov',50))
)

#ou importe o banco de dados
library(rio)
psi_df <- import('pasta_de_arquivos/psi_df.txt')
```

Primeiramente, vamos pedir ao R para calcular as frequências brutas e relativas, usando as funções table() e prop.table(), respectivamente. O parâmetro "margin = 1" serve para que a função prop.table() retorne as frequências relativas por linha. Caso você deseje por coluna, mude para o valor 2. O que mudará de fato é o jeito de contar a história. Mais uma vez round() entrará para reduzir o número de casas decimais.

```
#frequências absolutas
table(psi_df$personalidade,psi_df$metodo)
         Prat Teor
  Extrov   30   20
  Introv   10   40


#frequências relativas
round(prop.table(table(psi_df$personalidade,psi_df$metodo),margin = 1),2)
         Prat Teor
  Extrov 0.6 0.4
  Introv 0.2 0.8
```

Só olhando as frequências, nós podemos tirar diversas conclusões. Perceba que há 30 alunos que se consideram extrovertidos que se dão melhor com abordagens de ensino práticas, isso representa 60% entre os extrovertidos. Por outro lado, há 40% de extrovertidos que dão preferência

a abordagens teóricas (20 alunos). Dentre os que se consideram introvertidos, a discrepância é maior. Perceba que 80% (40 alunos) introvertidos preferem abordagem teórica, enquanto que apenas 20% (10 alunos) preferiram abordagem voltada para prática.

Em suma, parece que os alunos introvertidos gostam mais do método teórico, enquanto que os alunos extrovertidos gostam mais do método prático. Só que o mero olhar para as frequências é parte da análise. Para fechar uma conclusão sobre essa associação, vamos pedir para o R calcular o qui-quadrado de independência. Porém, antes de iniciarmos, precisamos falar sobre uma certa desvantagem do qui-quadrado: ele não é tão robusto assim, principalmente quando o n amostral é pequeno (já falamos sobre isso). A chance de encontrarmos associação, ou seja, p-valor $\leq 0{,}05$, é grande. Existem ajustes que deixam o qui-quadrado mais conservador e, portanto, mais rigoroso para rejeitarmos H_0. Um desses métodos é a correção de Yates.

Para calcular o qui-quadrado de independência, vamos usar ainda a função chisq.test(). Mudaremos o parâmetro "correct" para TRUE quando desejamos usar a correção de Yates, e FALSE quando não desejamos usar a correção. Vamos começar pelo qui-quadrado sem o ajuste de Yates.

```
#sem correção de Yates
chisq.test(psi_df$metodo,psi_df$personalidade, correct = FALSE)

      Pearson's Chi-squared test

data:  psi_df$metodo and psi_df$personalidade
X-squared = 16.667, df = 1, p-value = 4.456e-05
```

Podemos perceber que o p-valor foi significativo, ou seja, muito baixo. Rejeitaremos H_0, portanto. Então, concluímos que há associação entre o método de aprendizagem e o perfil de personalidade dos alunos.

```
#com correção de Yates
chisq.test(psi_df$metodo,psi_df$personalidade, correct = TRUE)

      Pearson's Chi-squared test with Yates' continuity correction

data:  psi_df$metodo and psi_df $personalidade
X-squared = 15.042, df = 1, p-value = 0.0001052
```

Pela correção de Yates podemos perceber que há a mesma associação, porém, o valor de P é um pouco maior, mas ainda significativo. Ou seja, ele foi mais rigoroso ao rejeitar H_0.

Boa parte dos pesquisadores não usam a correção de Yates por falta de conhecimento. Então, é muito mais comum usarem qui-quadrado ou o teste exato de Fisher como alternativa ao primeiro.

Com os resultados de frequências e o valor de P em mãos, podemos resumir os achados em uma tabela, como a que se segue:

Tabela 12.1. Associação entre os métodos de aprendizagem e o perfil de personalidade de alunos.

	Métodos de aprendizagem		
Perfil de personalidade	**Prático N (%)**	**Teórico N (%)**	**P[1]**
Extrovertido	30 (60,00)	20 (40,00)	< 0,001
Introvertido	10 (20,00)	40 (80,00)	

[1]Teste de qui-quadrado.

12.3 TESTE EXATO DE FISHER

Como visto acima, o teste de qui-quadrado tem algumas limitações. Para amostras pequenas, talvez a maior limitação seja a necessidade de ter pelo menos 5 observações em cada célula da nossa tabela. O teste exato de Fisher é uma alternativa usada para verificar associação entre variáveis categóricas em tabelas 2x2, quando as limitações do qui-quadrado ocorrerem.

Esse teste pode ser realizado com qualquer tamanho amostral, mas a sua limitação está em ser aplicado em apenas variáveis dicotomias. Seu modelo matemático foge do escopo deste livro, mas ele pode ser facilmente aplicável no R utilizando a função fisher.test(). Vamos ver isso usando o banco de dados "psi_df" trabalhado acima.

```
fisher.test(psi_df$metodo,psi_df$personalidade)
      Fisher's Exact Test for Count Data

data:  psi_df$metodo and psi_df$personalidade
p-value = 8.309e-05
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
  2.265787 16.390918
sample estimates:
odds ratio
  5.881258
```

Como resultado, o R retorna primeiramente o valor de P, que foi parecido com o do teste qui-quadrado sem ajuste de Yates. Depois disso, ele retorna a estatística da chance de risco (ou *odds ratio* – OR) que vimos no capítulo sobre regressão logística. Ele revela tanto o valor da OR quanto seu intervalo de confiança: 5,88 (2,27-16,39). A função que calcula o qui-quadrado não retorna a OR e seu IC, mas se você desejar esses resultados, sugiro trabalhar com a regressão logística e a mensuração de associação que aprendemos em capítulos anteriores.

Fechando a tabela de resultados obtidos, poderíamos expor da seguinte forma:

Tabela 12.2. Associação entre os métodos de aprendizagem e o perfil de personalidade de alunos.

	Métodos de aprendizagem			
	Prático	**Teórico**		
Perfil de personalidade	**N (%)**	**N (%)**	**P[1]**	**OR (IC)**
Extrovertido	30 (60,00)	20 (40,00)	< 0,001	5,88 (2,27-16,39)
Introvertido	10 (20,00)	40 (80,00)		

[1]Teste exato de Fisher.

Vamos ver outro exemplo, mais específico para o uso do teste exato de Fisher. Vamos supor que um pesquisador quer entender a relação entre o consumo de alimentos embutidos e sua relação com o câncer de cólon. Vamos dispor os dados no R e verificar as frequência brutas usando table():

```
df_ca <- data.frame(
  cancer = c(rep('Sim', 10), rep('Nao', 20)),
  consumo = c(rep('Sim',6), rep('Não', 21),rep('Sim',3))
)

#ou importe o banco
library(rio)
df_ca <- import('pasta_de_arquivos/df_ca.txt')

#tabela de frequência bruta
table(df_ca$consumo, df_ca$cancer,
      dnn = c('Embutidos','Câncer'))
         Câncer
Embutidos Nao Sim
      Não  17   4
      Sim   3   6
```

Perceba que nesse banco de dados há 30 observações; um n relativamente pequeno. Temos então dois problemas visíveis na distribuição de frequências, pois há duas células com valores menores do que 5 observações. Vamos usar prop.table() para analisarmos as proporções.

```
#tabela de frequência relativa
round(prop.table(table(df_ca$consumo, df_ca$cancer,
                       dnn = c('Embutidos','Câncer')),margin = 1),2)
         Câncer
Embutidos  Nao  Sim
      Não 0.81 0.19
      Sim 0.33 0.67
```

A partir disso, percebemos que a frequência de quem não come embutidos normalmente não apresentou câncer do cólon. Enquanto que quem consome esse tipo de alimento, a frequência é maior dentre quem tem tal doença (lembre que são dados fictícios).

O teste qui-quadrado aqui não é seguro, mas vamos calcular tanto ele quanto o exato de Fisher e observar seus resultados.

```
#teste X^2
chisq.test(df_ca$consumo, df_ca$cancer, correct = F)

      Pearson's Chi-squared test

data:  df_ca$consumo and df_ca$cancer
X-squared = 6.4286, df = 1, p-value = 0.01123

Warning message:
In chisq.test(df_ca$consumo, df_ca$cancer) :
  Aproximação do qui-quadrado pode estar incorreta
```

Perceba que o R retornou uma observação (*Warning message*) por conta das limitações que são impostas ao qui-quadrado. Pelo resultado de P, estaríamos aceitando que há associação entre alimentar-se com embutidos e a ocorrência de câncer de cólon (P = 0,011). Vamos ver agora o que o exato de Fisher retorna como resultado.

```
#exato de Fisher
fisher.test(df_ca$consumo, df_ca$cancer)

      Fisher's Exact Test for Count Data

data:  df_ca$consumo and df_ca$cancer
p-value = 0.03013
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
  1.106821 71.899878
sample estimates:
odds ratio
  7.760438
```

Pois bem. Temos um valor de P ainda significativo, porém maior do que o gerado pelo qui-quadrado. Ou seja, o teste exato de Fisher foi mais conservador. Geralmente a correção de Yates tende a ter resultados mais aproximados do teste exato de Fisher. Mas perceba que ainda a mensagem de alerta é retornada:

```
chisq.test(df_ca$consumo, df_ca$cancer, correct = T)
      Pearson's Chi-squared test with Yates' continuity correction


data:  df_ca$consumo and df_ca$cancer
```

```
X-squared = 4.4643, df = 1, p-value = 0.03461


Warning message:
In chisq.test(df_ca$consumo, df_ca$cancer, correct = T) :
  Aproximação do qui-quadrado pode estar incorreta
```

Resumindo essas análises, podemos dizer que se alimentar com embutidos tem a chance 7 vezes maior de ter câncer do cólon, segundo à pesquisa fictícia. Veja tabela de resultados:

Tabela 12.3. Associação entre os métodos de aprendizagem e o perfil de personalidade de alunos.

	Câncer de colón			
	Não	**Sim**		
Consumo de embutidos	**N (%)**	**N (%)**	**P[1]**	**OR (IC)**
Não	17 (81,00)	4 (19,00)	0,03	7,76 (1,11-71,90)
Sim	3 (33,00)	6 (67,00)		

[1]Teste exato de Fisher.

Uma coisa importante a se observar na Tabela 12.3 é o intervalo de confiança da OR. Perceba que ele tem uma amplitude bastante elevada, tornando-o impreciso. É um dos tantos problemas enfrentados quando temos um n amostral limitado.

Capítulo 3

Testes de Comparação de Médias

Vamos observar aqui um fato: temos médias de troponina sérica (um marcador bioquímico que pode avaliar infarto agudo do miocárdio) de dois grupos diferentes: A e B. A média para o grupo A foi 16,3 ng/L e a média do grupo B foi 16,3 ng/L. Essas duas médias são diferentes? Obviamente não, correto?

Um outro curioso dosou troponina em outro grupo, o C, que deu média de 16,5 ng/L. Um pouco semelhante ao grupo A, não? Mas também já há diferença.

Outro inquieto dosou a troponina em outros indivíduos, o grupo D, que apresentou média de 17,5 ng/L. Depois inventaram dosar em mais um grupo, o E, onde obtiveram a média 38,4ng/L. Parece que começou a ficar diferente demais.

A grande questão aqui é que quando estamos tratando variáveis numéricas, é muito raro que os valores absolutos de média, mediana, variância, desvio padrão e outras métricas sejam exatamente os mesmos valores em dois grupos, por mais semelhantes que possam parecer. Já que esses valores geralmente não são exatamente os mesmos entre os grupos em comparação, será que podemos considerar sempre que a média maior de um grupo é estatisticamente maior que o outro?

Se compararmos os grupos A e C apenas pelas suas médias, parecem muito semelhantes no quesito troponina, pois a diferença é meramente 0,2 ng/L. Por mais que os valores não sejam iguais, me passa a sensação de que ambos os grupos são bem parecidos. Entretanto, a comparação entre A e E é mais preocupante, pois se confrontarmos os valores de referência para a troponina (que para homens deve ser menor do que 16,8ng/L) o grupo E pode realmente estar em risco, pois sua média passou desse limite. E perceba o que mais importa nessa sessão, a diferença das médias entre A e E foi a maior entre as demais. Então, a nível de saúde cardíaca, nos parece que o grupo A goza de boa saúde cardíaca enquanto que o grupo E tem seus problemas.

Nos resta pensar agora sobre as médias dos grupos A e D. Apesar de D também apresentar um certo risco, será que é suficiente para eu afirmar com precisão que as médias entre A e D são suficientemente diferentes?

A questão aqui é: precisamos de um parâmetro que nos indique, de forma confiável, que uma média seja ou não estatisticamente diferente de outra. Para isso existem os testes de hipótese para comparação de médias na estatística. Contudo, não há um só método. Existem diferentes testes para diferentes necessidades. Acredito que esse capítulo seja um dos fundamentais da

bioestatística, uma vez que os pesquisadores estão quase que sempre comparando médias entre grupos em seus trabalhos.

À medida que formos apresentando os diferentes testes de comparação de médias, iremos mostrar as hipóteses testadas. Entretanto, genericamente podemos dizer que estamos testando as seguintes hipóteses:

H_0: as médias são iguais.

H_1: as médias são diferentes.

Vamos tomar sempre como base de decisão o resultado do p-valor calculado pelo teste e usaremos o nível de significância de 5%, então:

- p-valor $\leq 0{,}05$, rejeitar $H_0 \rightarrow$ teste significativo (médias diferentes).
- p-valor $> 0{,}05$, aceitar $H_0 \rightarrow$ teste não significativo (médias iguais).

Seguindo nossa jornada, é preciso entender quando uma pesquisa gera amostras independentes e quando gera amostras pareadas. Primeiramente, amostras independentes surgem quando os valores gerados de uma população não são relacionados, ou combinados em algum grau, com os valores de uma outra população. Normalmente, estamos diante de amostragem entre grupos de pessoas diferentes, analisando dados de máquinas diferentes, cepas de microrganismos diferentes, doentes e sadios, enfim. Já a amostragem pareada, existem um grau de relação ou combinação de características intrínsecas entre as duas amostragens. Normalmente são levantamentos de dados realizados na mesma pessoa em momentos diferentes: antes e depois. Podemos pensar também o levantamento de dados antes e depois de uma máquina ser ajustada ou calibrada.

Vamos exemplificar de forma mais prática. O zolpidem é um fármaco hipnótico muito utilizado para tratar insônia. É do grupo das imidazopiridinas e se assemelham aos benzodiazepínicos. Segundo Greenblatt e colaboradores (2014), há diferenças farmacocinéticas e farmacodinâmicas entre homens e mulheres. Para chegar a conclusões nesse sentido, é possível avaliar a dosagem desse fármaco após um tempo de administração e verificar se há diferenças nas médias dessas dosagens entre os sexos. É uma abordagem **independente**, pois os indivíduos têm estruturas e metabolismos relativamente diferentes.

Agora vamos para uma segunda situação. Pensemos em avaliar se a dosagem de 10mg do zolpidem é efetiva em mulheres, ou seja, os sinais de insônia melhoraram segundo um escore de 0 a 10. Se ao chegar ao médico uma paciente diz que seu problema de insônia tem escore 9 e depois

do tratamento ela afirma que seu escore agora é 3, é bem possível que ela tenha melhorado sua qualidade de vida. Isso é uma abordagem **pareada**, pois estamos avaliando os mesmos indivíduos em momentos diferentes do estudo.

Um outro fator que é importantíssimo levar em consideração é a distribuição dos dados. Se os dados numéricos estiverem na distribuição normal, usaremos determinadas técnicas estatísticas para testar a hipótese de que as médias sejam ou não iguais (abordagens **paramétricas**). Caso as distribuições não estejam na normal, utilizaremos outras técnicas para testar geralmente a igualdade entre as medianas (abordagens **não paramétricas**).

Ainda há mais uma consideração final: quantas médias estamos comparando? Queremos saber se uma determinada média amostral é igual a uma populacional já conhecida, então usamos alguns métodos. Se são duas médias, então temos a disposição alguns outros métodos. Se são mais de duas médias, então a abordagem será diferente. A Figura 13.1 traz um fluxograma de como devemos escolher o método estatístico correto para essa questão de comparação de médias. Ele simplifica a decisão de qual estatística devemos utilizar.

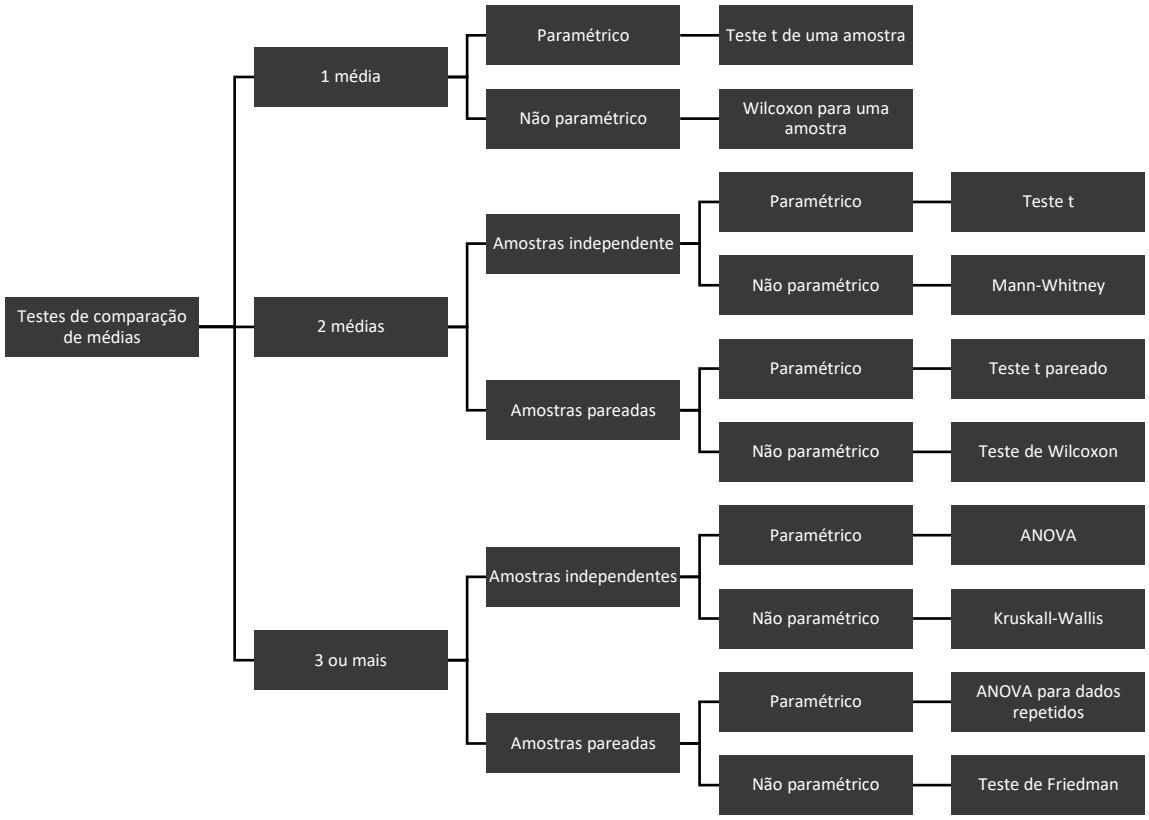


Figura 13.1 Fluxograma para o uso de testes de comparação de médias.

Cada um desses testes tem sua fórmula matemática específica. Se você deseja se aprofundar em alguma ou todas elas, sugiro que comece estudando a documentação de cada função que usaremos a seguir. Focaremos nas necessidades, meios de testar as hipóteses e suas conclusões.

13.1 COMPARAÇÃO DE UMA MÉDIA AMOSTRAL COM UMA MÉDIA POPULACIONAL CONHECIDA

Esses testes que iremos apresentar são comumente chamados de teste de uma amostra. Isso porque o pesquisador tem em mãos apenas uma amostra e deseja comparar sua média com uma média já bem estabelecida, seja pela literatura ou por outros meios de padronização.

Por exemplo, um gestor público de saúde está preocupado com a demora na fila de espera dos pacientes para atendimentos. Em pesquisas prévias em seu município, ele sabe que no passado a média de espera para atendimentos em Unidades de Pronto Atendimento (UPA) era de 40 minutos, com desvio padrão de 5 minutos.

Ele deseja saber se no primeiro ano do seu mandato houve diferença estatística em relação ao passado. Por isso, ele solicitou a uma equipe especializada que coletassem esses dados em uma determinada UPA de interesse. Tabularam os dados e avaliaram o teste estatístico adequado.

Falando do teste adequado em frente a esse problema, precisamos primeiramente entender o que se deseja: temos uma média e um desvio padrão já estabelecidos; temos um levantamento de dados (uma amostragem) e vamos fazer comparações do que já sei com a amostra que tenho em mãos. Depois, sendo um levantamento de dados numéricos, precisamos testar a normalidade. Caso os dados estejam na distribuição normal, usaremos o teste t de *Studant* (paramétrico) para uma amostra. Caso contrário, usaremos o teste não paramétrico de Wilcoxon para uma amostra.

Vamos escrever as hipóteses estatísticas para o problema:

H_0: a média amostral é igual a 40 minutos.

H_1: a média amostral é diferente de 40 minutos.

Vamos criar ou importar o banco de dados, que supostamente a equipe do gestor tabulou recolhendo dados de tempo entre a chegada e o atendimento médico de mil indivíduos que foram atendidos em um determinado mês em uma UPA que ele está avaliando.

```
#criação do banco de dados
set.seed(1234)
df1 <- data.frame(
```

```
  tempos = round(rnorm(1000, mean = 36, sd = 11),2)
)

#ou importar arquivo
library(rio)
df1 <- import('pasta_de_arquivos/df1.txt')

#visualizando o banco
head(df1)
  tempos
1  22.72
2  39.05
3  47.93
4  10.20
5  40.72
6  41.57

#Teste de normalidade
shapiro.test(df1$tempos)

   Shapiro-Wilk normality test

data:  df1$tempos
W = 0.99737, p-value = 0.1052
```

Testamos a normalidade com o teste de Shapiro-Wilk. Pelo p-valor retornado (P > 0,05; não significativo), concluímos nessa etapa que a distribuição dos dados está na normal. Por isso, seguimos com **teste t paramétrico para uma amostra**. Para comparar se a média amostral é igual ou não da população, usaremos a função t.test() e seu parâmetro "mu" que deve conter a média esperada (populacional). Tenha cuidado nesse momento, pois se esquecermos de preencher esse parâmetro, o R considerará a média esperada como zero, como vem como padrão.

```
#teste t para uma amostra
t.test(df1$tempos, mu = 40)

      One Sample t-test

data:  df1$tempos
t = -12.373, df = 999, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 40
95 percent confidence interval:
 35.02664 36.38822
sample estimates:
mean of x
 35.70743
```

O R retorna primeiramente o tipo de teste: *One Sample t-test*. O R calculou a estatística t (t = -13,299), utilizou como graus de liberdade igual a 999, pois graus de liberdade nesse teste é igual a n-1. Seu p-valor foi significativo e podemos resumi-lo da seguinte maneira: P < 0,001. Sendo assim, rejeitamos H_0. Perceba que o R escreve a hipótese alternativa "alternative hypothesis:

true mean is not equal to 40". Então, concluímos que a média amostral é diferente de 40 minutos. Especificamente, ela é menor do que a média de espera nos atendimentos do passado. É possível que a equipe gestora da UPA tenha melhorado a logística de atendimento aos pacientes, já que a média de espera para essa UPA foi de 35,71 (intervalo de confiança da média: 35,03-36,39) minutos.

Caso esse p-valor fosse maior que 0,05, consideraríamos que a média amostral é igual a 40 min, que é a média populacional esperada. Como conclusão, não haveria qualquer efeito eficiente da nova equipe em relação ao tempo de espera para os atendimentos.

Vamos sopor agora que o mesmo gestor coletou uma amostra de um determinado hospital e testar a mesma hipótese anterior. Normalmente a entrada de paciente nos hospitais é de caráter mais complexo. É esperado que o tempo varie mais, portanto. Vamos aos dados fictícios de "df2" e testar a normalidade.

```
set.seed(1234)
df2 <- data.frame(
  tempos = round(rlnorm(1000, meanlog = 5, sdlog = 2),2)
)

#ou importando dados
df2 <- import('pasta_de_arquivos/df2.txt')


#visualisando dados
head(df2)
   tempos
1   13.27
2  258.49
3 1298.39
4    1.36
5  350.11
6  408.34


#testando a normalidade dos tempos de atendimento
shapiro.test(df2$tempos)

    Shapiro-Wilk normality test

data:  df2$tempos
W = 0.19275, p-value < 2.2e-16
```

Podemos observar que o p-valor foi significativo para esse teste de normalidade, portanto, essa distribuição está fora da normal. Devemos seguir com o **teste não paramétrico de Wilcoxon para uma amostra**. Usaremos a função wilcox.test() e nela devemos também utilizar o parâmetro "mu". Contudo, uma observação prévia deve ser feita: quando as distribuições não estão na normal,

damos preferência à comparação de medianas. Se o pesquisador tiver acesso à mediana da população, é interessante que ele utilize aqui. Vamos supor que nesse exemplo hipotético a média e mediana da população se equivalem por serem muito próximas.

```
#Teste de Wilcoxon para uma amostra
wilcox.test(df2$tempos, mu = 40)

      Wilcoxon signed rank test with continuity correction

data:  df2$tempos
V = 446010, p-value < 2.2e-16
alternative hypothesis: true location is not equal to 40
```

Como resultado do teste de Wilcoxon para uma amostra, há diferença estatística entre as medianas, uma vez que o valor de P foi significativo. Vamos verificar a mediana amostral:

```
median(df2$tempos)
[1] 137.06
```

A mediana da nossa amostragem é bem diferente da mediana esperada. Concluímos que as medianas são estatisticamente diferentes, com maior tempo de espera nos hospitais.

13.2 COMPARAÇÃO DE DUAS MÉDIAS DE AMOSTRAGENS INDEPENDENTES

As amostras independentes tratam de indivíduos de grupos diferentes e especificamente aqui trabalharemos na comparação de dois grupos. Por exemplo, queremos avaliar o comprimento médio da circunferência abdominal entre homens e mulheres; avaliar a diferença da quantidade média de leucócitos entre pacientes saudáveis e com sepse; avaliar a quantidade média de colônias que crescem em meio de cultura com e sem um determinado fármaco.

Também aqui, avaliar a normalidade dos dados é importante. Mas, não é adequado avaliar a normalidade de todos os dados juntos. Devemos testar a normalidade para cada grupo separadamente. Vamos entender essa diferença na prática a seguir.

Vamos construir ou importar um banco de dados o qual um pesquisador deseja comparar o antígeno CA 15-3, um marcador para câncer de mama comumente solicitado por oncologistas. Ele deseja comparar os níveis do CA 15-3 (em U/mL) no sangue de pacientes controle (mulheres jovens sem qualquer sinal prévio de nódulos na mama) e pacientes caso (mulheres de idade mais avançada, presença de nódulo identificado por mamografia). Chamaremos no banco de “df3”.

```
#montar o banco hipotético
```

```
set.seed(1111)
df3 <- data.frame(
  grupo = c(rep("controle",50),rep('caso',50)),
  CA153 = c(round(c(rnorm(50,mean=20),rnorm(50,mean=90)),2))
)

#ou importando dados
df3 <- import('pasta_de_arquivos/df3.txt')


head(df3)
     grupo CA153
1 controle 19.91
2 controle 21.32
3 controle 20.64
4 controle 21.17
5 controle 20.12
6 controle 17.07
```

Se você rodar o comando abaixo, os resultados que serão gerados estarão equivocados em relação a nosso propósito. Veja se você consegue identificar o erro.

```
#calculando a média
mean(df3$CA153)
[1] 55.2411

#rodando teste de normalidade
shapiro.test(df3$CA153)

   Shapiro-Wilk normality test

data:  df3$CA153
W = 0.66561, p-value = 9.139e-14
```

Primeiramente, a média que calculamos é a média geral dos indivíduos. Pouco nos é informativo, já que temos dois grupos bem definidos. Além do mais, ao realizado o teste de normalidade, estamos cometendo o erro de testar de forma geral também.

Então, vamos calcular a média de cada grupo, usando como auxílio a função subset() para filtrar a classe que se deseja calcular a média.

```
mean(subset(df3, grupo == "controle")$CA153)
[1] 20.1986

mean(subset(df3, grupo == "caso")$CA153)
[1] 90.2836
```

Observemos que a média sérica do marcador é numericamente maior entre os casos. Vamos testar agora corretamente a normalidade da distribuição de dados em cada grupo.

```
#teste de normalidade nos controles
shapiro.test(subset(df3, grupo == "controle")$CA153)
```

```
   Shapiro-Wilk normality test

data:  subset(df3, grupo == "controle")$CA153
W = 0.98985, p-value = 0.9427


#teste de normalidade nos casos
shapiro.test(subset(df3, grupo == "caso")$CA153)

   Shapiro-Wilk normality test

data:  subset(df3, grupo == "caso")$CA153
W = 0.95919, p-value = 0.08222
```

Segundo os dois testes de normalidade, as distribuições encontram-se na normal, pois ambos p-valores foram maiores do que 0,05 (diferente do que o teste com todos os dados juntos havia sugerido, perceba).

Como as duas distribuições estão na normal, seguiremos com o teste t de comparação de médias independentes. Caso uma das distribuições, ou as duas, estivessem fora da normal, seguiríamos com o **teste não paramétrico de Mann-Whitney**, que comentaremos mais à frente.

Contudo, ainda há um ponto em que muitos pesquisadores escorregam. Ao fazer o teste t, eles não avaliam um pré-teste que pode ser fundamental para a veracidade do resultado. Antes de correr o teste t, precisamos avaliar se as variâncias dessas distribuições são iguais ou diferentes, porque há um teste t específico para quando são iguais e outro para quando são diferentes. Essa avaliação prévia é chamada de homocedasticidade, também conhecida como homogeneidade de variâncias. É também muito comum seu uso nas regressões e ANOVA (que veremos mais à frente).

Para avaliar a homocedasticidade, vamos usar a função var.test(), que usa a estatística F para avaliar essa igualdade. Nela, precisamos atribuir valor aos parâmetros x e y, que são os valores numéricos de cada grupo. Essa primeira hipótese a ser testada será:

H_0: as variâncias entre os grupos são iguais.

H_1: as variâncias são diferentes.

Veja na Figura 13.2 o fluxograma simples para seguirmos com o teste t.

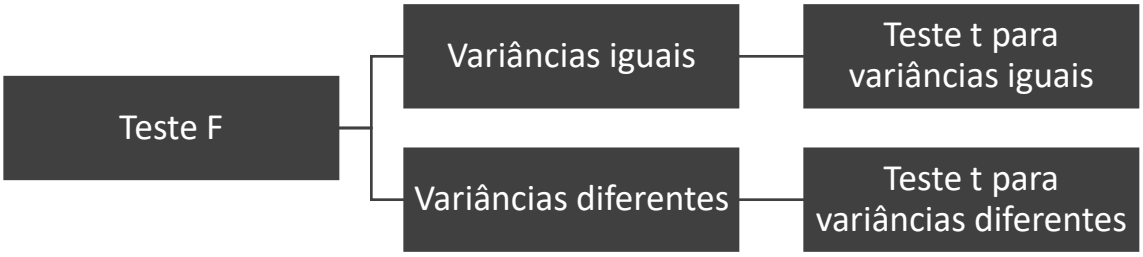


Figura 13.2. Esquema para realizar o teste t de *Studant* para amostras independentes.

```
#teste F
var.test(subset(df3, grupo == "controle")$CA153,
         subset(df3, grupo == "caso")$CA153)

      F test to compare two variances

data:  subset(df3, grupo == "controle")$CA153 and subset(df3, grupo == "caso")$CA153
F = 1.3091, num df = 49, denom df = 49, p-value = 0.349
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.7429026 2.3069417
sample estimates:
ratio of variances
          1.309134
```

O resultado do teste F nos indica que as variâncias são estatisticamente iguais, já que o seu p-valor não foi significativo, portanto, aceitamos H_0.

Para realizar o teste t de *Studant* de amostras independentes usaremos a função t.test(). Há duas formas de entrada de dados nessa função: (1) podemos pôr como fórmula [variável numérica ~ variável categórica]; (2) ou, x e y, onde x é o conjunto de valores numéricos de um grupo e y é o conjunto numérico do segundo grupo. Ainda, nessa função devemos especificar se as variâncias são iguais ou diferentes pelo parâmetro "var.equal". Sendo as variâncias estatisticamente iguais, coloque TRUE, caso contrário, coloque FALSE.

As hipóteses do teste t de *Studant* serão:

H_0: as duas médias são iguais.

H_1: as duas médias são diferentes.

```
t.test(df3$CA153 ~ grupo, data = df3, var.equal = TRUE)

      Two Sample t-test

data:  df3$CA153 by grupo
t = 316.07, df = 98, p-value < 2.2e-16
```

```
alternative hypothesis: true difference in means between group caso and group con-
trole is not equal to 0
95 percent confidence interval:
 69.64497 70.52503
sample estimates:
    mean in group caso mean in group controle
               90.2836                20.1986
```

Como podemos observar, o p-valor gerado é menor do que 0,05, indicando que as médias são estatisticamente diferentes, pois rejeitamos H_0. Perceba ainda que o R retorna as duas médias e isso é um bom ponto de checagem para a gente.

Vamos agora exemplificar quando as distribuições não estão na normal. Usaremos o mesmo exemplo acima, mas vamos mudar sensivelmente os dados, vamos chamar o banco de "df4".

```
set.seed(2222)
df4 <- data.frame(
  grupo = c(rep("controle",50),rep('caso',50)),
  CA153 = c(round(c(rnorm(60,mean=20),rnorm(40,mean=90)),2))
)

#ou importe df4
library(rio)
df4 <- import('pasta_de_arquivos/df4.txt')

head(df4)
     grupo CA153
1 controle 19.66
2 controle 20.94
3 controle 21.74
4 controle 20.70
5 controle 20.46
6 controle 19.68
```

Vamos pedir a R para testar as distribuições usando Shapiro-Wilk.

```
shapiro.test(subset(df4, grupo == "controle")$CA153)

      Shapiro-Wilk normality test

data:  subset(df3, grupo == "controle")$CA153
W = 0.98641, p-value = 0.8303


shapiro.test(subset(df4, grupo == "caso")$CA153)

      Shapiro-Wilk normality test

data:  subset(df3, grupo == "caso")$CA153
W = 0.52164, p-value = 1.572e-11
```

No banco "df4", percebermos que a distribuição do marcador sérico está fora da normal entre os casos. Isso já é suficiente para que o teste de comparação de médias seja não paramétrico. Portanto, vamos rodar a análise de Mann-Whitney. Mas antes disso, vamos analisar graficamente as duas distribuições, para confrontarmos com a estatística gerada acima. Vemos através da Figura 13.3 que realmente a distribuição de CA15-3 sérica entre os controles está muito semelhante à normal. Contudo, a distribuição entre os casos há uma disparidade bem evidente, completamente distinta da distribuição normal.

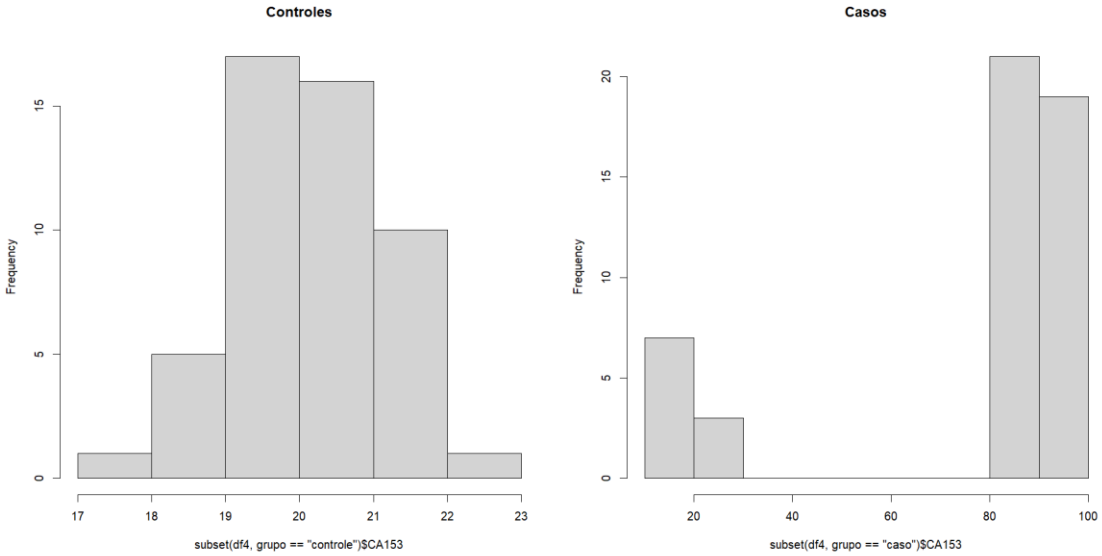


Figura 13.3. Histograma dos níveis de CA15-3 entre controles e casos.

As hipóteses estatísticas em jogo continuam sendo:

H_0: as médias de CA15-3 são iguais entre caso e controle.

H_1: as médias de CA15-3 são diferentes entre caso e controle.

No R, usaremos a função wilcox.test () para julgarmos H_0. Essa função é mais sensível no uso de fórmulas, podendo calcular variações do teste de Wilcoxon que não seja a nossa necessidade em questão. Por conta disso, é melhor usarmos a entrada de dados do tipo x e y (subdivisão dos dados).

```
wilcox.test(subset(df4, grupo == "controle")$CA153,
            subset(df4, grupo == "caso")$CA153)

      Wilcoxon rank sum test with continuity correction

data:  subset(df4, grupo == "controle")$CA153 and subset(df4, grupo == "caso")$CA153
W = 337.5, p-value = 3.23e-10
alternative hypothesis: true location shift is not equal to 0
```

O resultado nos retorna um p-valor menor do que 0,05. Portanto, temos evidência significativas para rejeitamos H_0, indicando que as médias são estatisticamente diferentes.

13.3 COMPARAÇÃO DE DUAS MÉDIAS DE AMOSTRAS PAREADAS

Quando estamos testando um novo medicamento, por exemplo, geralmente nós avaliamos diversas variáveis antes e após o uso do mesmo. É de grande interesse que haja melhora na maioria dos casos, obviamente, mas precisamos avaliar também a ocorrência de efeitos adversos e colaterais.

É importante salientar que nesse tipo de trabalho estamos avaliando o mesmo indivíduo em momentos diferentes. Então, não tem como ser uma comparação independente, pois não há grupos independentes a serem comparados. O que há é a comparação da mesma variável, nos mesmos indivíduos, em momentos diferentes. Nesse meio, quando temos uma variável numérica, podemos realizar comparações de médias ditas pareadas.

Como sempre, a distribuição dos dados é importante avaliar. Há métodos diferentes a serem usados, quando a distribuição está na normal (t**este t pareado**) e quando não está na normal (**teste de Wilcoxon pareado**). Para que não fique repetitivo, vamos sempre testar antes a normalidade com o teste de Shapiro-Wilk de forma mais direta de agora em diante.

Vamos a exemplos. Existem alguns diuréticos, como a hidroclorotiazida, que são utilizados para tratar pacientes com hipertensão. Um pesquisador deseja testar o efeito da pressão arterial sistólica (PAS) utilizando um novo fármaco da mesma classe da hidroclorotiazida. Ele selecionou 30 pacientes com histórico de hipertensão, os quais todos tomaram esse novo fármaco e na mesma concentração. Nessa fase de testes, o pesquisado deseja avaliar apenas se há diferença da pressão sanguínea antes e após o uso desse novo fármaco.

Os dados fictícios vamos chamar de “df5” e a hipótese a ser testada aqui será:

H_0: as médias de PAS de antes e depois da administração do medicamento são iguais.
H_1: as médias de PAS antes e depois da administração do medicamento são diferentes.

Só por curiosidade, esse banco de dados pode ser feito de duas formas básicas diferentes. O primeiro, podemos atribuir vetores diferentes entre os dois momentos: antes e depois.

```
Antes <- c(1,2,3,4,5)
Depois <- c(2,3,4,5,6)
```

Ou, atribuir valores a duas variáveis de um banco de dados: uma categoria definindo o momento da coleta de dados e a outra numérica com a variável que se deseja calcular as médias.

```
banco_de_dados <- data.frame(
  Var_de_agrupamento = c('Antes','Antes','Depois','Depois'),
  Var_numerica = c(1,2,3,4)
)
```

Se estiver trabalhando com poucos dados, um teste rápido sem muito compromisso, use o primeiro modelo sem medo. Entretanto, se você estiver trabalhando com uma quantidade de dados maior, a segunda forma é melhor, pois é o padrão da maioria das tabulações que são feitas em pesquisa. Ou seja, um objeto único, onde cada coluna é uma variável, onde cada variável nos traz uma informação. Se você prestar atenção, no primeiro modelo temos dois objetos, que cada um é na verdade uma variável que traz a mesma informação (pressão sistólica, no nosso exemplo).

Dito isso, vamos construir um banco fictício e seguir com nossas análises de normalidade dos dados.

```
set.seed(777)
df5 <- data.frame(
  momento = c(rep("Antes",30),rep("Depois",30)),
  PAS = round(rnorm(60,160,10))
)

#ou importe os dados
library(rio)
df5 <- import('pasta_de_arquivos/df5.txt')

head(df5)
  momento PAS
1   Antes 165
2   Antes 156
3   Antes 165
4   Antes 156
5   Antes 176
6   Antes 166


#testando a normalidade em “Antes”
shapiro.test(subset(df5, momento == "Antes")$PAS)

   Shapiro-Wilk normality test

data:  subset(df5, momento == "Antes")$PAS
W = 0.97428, p-value = 0.6616


#testando a normalidade em “Depois”
shapiro.test(subset(df5, momento == "Depois")$PAS)

   Shapiro-Wilk normality test
```

```
data:  subset(df5, momento == "Depois")$PAS
W = 0.97604, p-value = 0.7134
```

Como resultado dos testes de normalidade, observamos que as duas distribuições estão na normal pois não houve p-valor significativo em ambos. Portanto, devemos seguir com teste paramétrico, especificamente o teste t para amostras pareadas. A função que utilizaremos para calcular o teste será ainda a t.test(), contudo, usaremos o parâmetro "paired = TRUE" para indicar ao R que desejamos realizar o teste pareado. Caso esse parâmetro estiver como FALSE (como ele vem como padrão) o R calculará o teste t para amostras independentes. Portanto, atenção!

Vamos pedir ao R para retornar as médias (ainda usando a função subset() para dividir o banco em subgrupos) e depois testar se a média de antes é igual a média de depois do uso do suposto novo medicamento.

```
mean(subset(df5, momento == "Antes")$PAS)
[1] 160.9333

mean(subset(df5, momento == "Depois")$PAS)
[1] 161.0333

#teste t pareado
t.test(subset(df5, momento == "Antes")$PAS,
       subset(df5, momento == "Depois")$PAS,
       paired = TRUE)

      Paired t-test
data:  subset(df5, momento == "Antes")$PAS and subset(df5, momento == "Depois")$PAS
t = -0.040521, df = 29, p-value = 0.968
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -5.147393  4.947393
sample estimates:
mean difference
           -0.1
```

Como resultado, o t calculado foi um valor baixo próximo de zero (t = -0.040521), o que é um indício para que o teste não seja significativo. Pelo p-valor não significativo (p-value = 0.968; P > 0,05), podemos dizer que o fármaco não tem efeito sobre a pressão arterial sistólica, pois as médias entre os momentos "Antes" e "Depois" são estatisticamente iguais.

Agora, vamos avaliar um outro banco de dados, o qual chamaremos de "df6". Nesse, vamos supor que o pesquisador tem os mesmos questionamentos sobre o efeito do novo medicamento sobre a PAS, mas não satisfeito com os primeiros resultados, ele realizou uma segunda amostragem.

```
set.seed(4444)
df6 <- data.frame(
  momento = c(rep("Antes",40),rep("Depois",40)),
  PAS = round(rlnorm(80,1)+150,2)
)

#ou importe os dados
library(rio)
df6 <- import('pasta_de_arquivos/df6.txt')

head(df6)
  momento    PAS
1   Antes 173.79
2   Antes 150.37
3   Antes 150.52
4   Antes 155.33
5   Antes 157.28
6   Antes 154.58
```

Testando a normalidade:

```
shapiro.test(subset(df6, momento == "Antes")$PAS)

      Shapiro-Wilk normality test

data:  subset(df6, momento == "Antes")$PAS
W = 0.4855, p-value = 1.153e-10


shapiro.test(subset(df6, momento == "Depois")$PAS)

      Shapiro-Wilk normality test

data:  subset(df6, momento == "Depois")$PAS
W = 0.46983, p-value = 7.636e-11
```

Pelos resultados prévios de normalidade, os p-valores foram ambos significativos. Dessa forma, concluímos até aqui que as distribuições de PAS não estão na normal; devemos seguir com um teste de comparação de médias pareado não paramétrico. Nesse caso específico, o teste de Wilcoxon para duas amostras pareado é o que satisfaz a nossa necessidade. Para isso, usaremos a função wilcox.test(). Nela, precisamos especificar o parâmetro "paired=TRUE" para informarmos que se refere a amostras pareadas.

Como se trata de um teste não paramétrico, a hipótese a ser testada aqui será a comparação de medianas:

H_0: os valores medianos de PAS entre antes e depois da administração do medicamento são iguais.

H_1: os valores medianos de PAS entre antes e depois da administração do medicamento são diferentes.

Vamos rodar o teste:

```
wilcox.test(subset(df6, momento == "Antes")$PAS,
            subset(df6, momento == "Depois")$PAS,
            paired = TRUE)

      Wilcoxon signed rank exact test

data:  subset(df6, momento == "Antes")$PAS and subset(df6, momento == "Depois")$PAS
V = 404, p-value = 0.9417
alternative hypothesis: true location shift is not equal to 0
```

Pelo resultado retornado, o p-valor não foi significativo (P > 0,05). Temos então evidências suficientes para aceitarmos H_0 e concluir que os valores medianos de PAS antes e depois são estatisticamente iguais. De fato, o novo medicamento não tem qualquer efeito na alteração da pressão sistólica.

13.4 DEMONSTRAÇÃO DE RESULTADOS PARA COMPARAÇÃO DE MÉDIAS

Finalizando a discussão sobre a comparação entre duas médias, agora independentemente se as médias são pareadas ou independentes, um bom auxílio é o uso gráfico para visualizar quão dispersas estão as médias ou medianas de cada grupo. Geralmente, usamos gráficos de barras para comparar médias (as distribuições estão na normal, portanto), enquanto que é mais conveniente usarmos o boxplot para comparar as medianas (distribuições fora da normal, portanto). O gráfico de barras tem o seu limite da barra o valor da média e normalmente acoplado a essa tem as barras de erro (erro da média ou desvio padrão), veja Figura 13.4-A. Lembre-se que o boxplot usa os quartis para compor o gráfico, cujo segundo é a mediana (Figura 13.4-B). Por conta disso, é justificável seu uso na comparação de medianas. Entretanto, isso não é exatamente uma regra. Particularmente, prefiro usar boxplot para ambas situações, evidenciando a média quando necessário (Figura 13.4-C).

Na Figura 13.4-A foram usados os dados do banco “df3”, cujas distribuições seguiam a normal e por isso usamos o teste t de *Studant*. Nesse gráfico é perceptível a discrepância entre as médias e confirmamos que há diferença estatística entre elas através do seu p-valor. As barras de erro utilizadas no gráfico equivalem ao desvio padrão, mas podemos utilizar também o erro padrão da média. Quando essas barras de erro estão muito afastadas (com base no eixo y), é um indício visual de que há diferença estatística.

Na Figura 13.4-B foram usados os dados de "df6", que foi uma comparação pareada (antes x depois), mas cujas distribuições estavam fora da normal, por isso o teste de Wilcoxon pareado. Dá para perceber que as medianas têm valores próximos (as linhas pretas dentro das caixas), mas só podemos afirmar que as medianas são estatisticamente iguais avaliando o p-valor, que não foi significativo. Aqui também existe um indício visual que sugere a igualdade entre as médias. Perceba que as caixas se sobrepõem em relação ao eixo y. Isso mostra pelo menos que 50% dos dados estão em um espectro muito próximo. Difícil haver diferença estatística nessas condições.

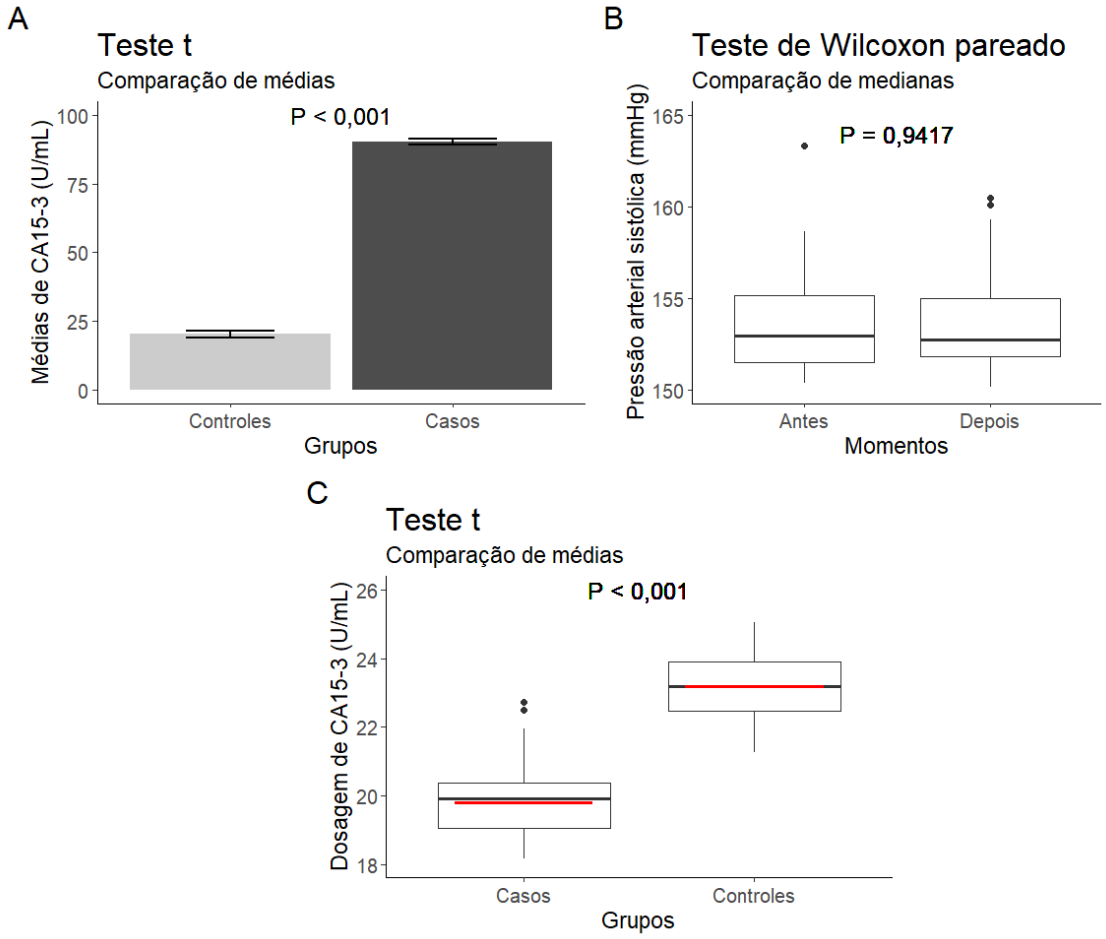


Figura 13.4. Formas diferentes de demonstrar graficamente resultados de comparação entre duas médias.

Na Figura 13.4-C fugimos à regra. São dados do banco "df3" modificado (redução dos dados entre os casos apenas para caber na imagem). As distribuições dos dados numéricos ainda estão na normal, o que nos leva ao teste t para compararmos as médias. Entretanto, o gráfico

mostrado é um boxplot, cujas médias estão destacadas em vermelho. Esse destaque é opcional, para ser transparente com o seu público, mas lembre-se que quando a distribuição dos dados estiver na normal, a média será muito próxima da mediana (isso é perceptível no gráfico, já que média e mediana praticamente se sobrepõem). Além do mais, perceba que não há sobreposição entre as caixas e isso é o indício visual de que há diferença estatística entre as médias.

Sobre a construção de cada gráfico, aprenderemos em um capítulo à parte. Da mesma forma com montagem e apresentação de tabelas. Mas por ora, a comparação de médias/medianas seria algo como a Tabela 13.1, que mostra resultados dos bancos "df" e "df". Quanto aos intervalos, para a média podemos usar ainda o erro padrão da média ou desvio padrão (que o mais comum). Para a mediana, é mais interessante expor o desvio interquartílico.

Tabela 13.1. Comparações das dosagens de CA15-3 sérico entre casos e controles.

	Grupo		
	Controles	**Casos**	**P**
CA 15-3 (U/mL)[1]	20,20±1,18	90,28±1,03	< 0,001
CA 15-3 (U/mL)[2]	20,09±1,33	89,52±2,13	< 0,001

[1]Amostragem 1: média ± desvio padrão, P: teste t; [2]Amostragem 2: mediana ± desvio interquartílico, P: teste de Mann-Whitney.

13.5 COMPARAÇÃO DE TRÊS OU MAIS MÉDIAS

A comparação de três médias ou mais médias é feita por técnicas relativamente diferentes das descritas até o momento. As preocupações pré-teste, diria assim, ainda são as mesmas. Primeiramente, ainda devemos testar a normalidade das distribuições. Segundo, nós devemos testar a hipótese de que as médias são estatisticamente iguais ou não. A diferença vem após essa fase, pois devemos realizar outros testes para saber quais médias são diferentes entre si, par a par. Essa parte final há inúmeros métodos e são chamados de testes post-hoc.

Na primeira abordagem, existem dois testes importantes, dependendo da normalidade dos dados, usamos a **Análise de Variância** (ANOVA) quando os eles estão na normal e o Kruskal-Wallis quando a distribuição não está na normal. Veremos cada um separadamente.

13.5.1 ANOVA (de amostras independentes)

A Análise de Variância, ou ANOVA (do inglês *ANalysis Of VAriance*), utiliza a estatística F e a distribuição F para testarmos a hipótese de que algumas entre as médias em estudo são diferentes entre si, desde que o pressuposto de que todas as distribuições estejam na normal seja verdadeiro.

Outro pressuposto aqui é que as amostras sejam independentes, pois a intensão nesse momento é comparar grupos independentes entre si. Essa ANOVA que iremos trabalhar pode ser chamada de ANOVA de um fator (*One-Way* ANOVA), pois teremos a variável dependente como numérica e a variável independente a variável categórica que classifica os grupos, que deve ser três ou mais. Essa variável categórica também podemos chamar de fator.

Existe só mais um pressuposto para ANOVA, que é a de homocedasticidade. Esse termo se refere à igualdade das variâncias da variável numérica entre os grupos analisados. Vimos avaliação semelhante nos testes acima, quando verificamos se as variâncias eram iguais ou diferentes entre os dois grupos, no uso do teste t independente. Na ANOVA convencional, precisamos de homocedasticidade. Quando não houver, devemos partir para outras alternativas de comparação de três ou mais médias: transformação dos dados (para logaritmo, normalização ou padronização), uso da ANOVA robusta ou teste não-paramétrico equivalente como o Kruskal-Wallis.

A avaliação da homocedasticidade pode ser realizada pelo teste de Levene. Podemos usar o pacote car (*Companion to Applied Regression*), com a função leveneTest() para isso. Desventuradamente, pouquíssimos pesquisadores realizam esse teste, talvez por falta de conhecimento do pressuposto de homogeneidade das variâncias na ANOVA, o que pode não ser culpa dos nossos colegas. Muitos livros básicos de estatística não adentram nesse assunto e acabam por não discutirem esse passo. Eu sugiro que você o realize em seus trabalhos, mas talvez não seja necessário expor seus resultados; fica a seu critério. Entretanto, o teste de Levene é um bom indicador de qual grupo de testes post-hoc melhor usar. Veremos isso logo mais à frente.

A hipótese a ser testada na ANOVA de um fator é:

H_0: todas as médias são iguais entre si.
H_1: há pelo menos uma média diferente das demais.

Exemplificando uma ANOVA convencional, imagine que um cientista se interesse em comparar a taxa sérica de glicose entre pacientes controlados, não controlados e que ainda não sabem do diagnóstico de diabetes melitos. Para eles construir seu banco de dados, ele terá que criar

duas variáveis, minimante: (1) dosagem sérica de glicose; (2) grupo. A dosagem de glicose é, portanto, a variável dependente e a variável grupo é a variável independente (fator).

Vamos construir um banco chamado "df7". Nele, iremos atribuir letras para os grupos: A = pacientes com diabetes controlada; B = pacientes com diabetes não controlada; C = pacientes diabéticos que ainda não sabem o diagnóstico.

```
set.seed(123)
df7 <- data.frame(
  grupo = as.factor(rep(c("A", "B", "C"), each = 50)),
  glicose = round(c(A = rnorm(50,90,10),# glicose do grupo A
                    B = rnorm(50,150,11), # glicose do grupo B
                    C = rnorm(50,145,12) # glicose do grupo C
  ),2)
)

#ou importe os dados
library(rio)
df7 <- import('pasta_de_arquivos/df7.txt')

head(df7)
   grupo glicose
A1     A   84.40
A2     A   87.70
A3     A  105.59
A4     A   90.71
A5     A   91.29
A6     A  107.15
```

Vamos testar a normalidade entre os três grupos usando Shapiro-Wilk.

```
shapiro.test(subset(df7, grupo == "A")$glicose)

      Shapiro-Wilk normality test
data:  subset(df7, grupo == "A")$glicose
W = 0.98927, p-value = 0.9278


shapiro.test(subset(df7, grupo == "B")$glicose)

      Shapiro-Wilk normality test
data:  subset(df7, grupo == "B")$glicose
W = 0.99075, p-value = 0.962


shapiro.test(subset(df7, grupo == "C")$glicose)

      Shapiro-Wilk normality test
data:  subset(df7, grupo == "C")$glicose
W = 0.97211, p-value = 0.2813
```

Segundo os resultados, as três distribuições estão na normal, pois nenhum p-valor foi significativo (P > 0,05). Dessa forma, podemos seguir com a ANOVA. Caso alguma ou todas as distribuições estivessem fora da normal, seguiríamos com o **teste Kruskal-Wallis** (não-paramétrico) que aprenderemos mais à frente.

Agora, devemos testar a homocedasticidade. Vamos instalar o pacote car e usar a sua função leveleTest() da seguinte maneira:

```
#teste de homocedasticidade
install.packages('car')
library(car)

leveneTest(glicose ~ grupo, data = df7)
Levene's Test for Homogeneity of Variance (center = median)
       Df F value Pr(>F)
group   2  1.3829 0.2541
      147
```

Como resultados, podemos afirmar que há homocedasticidade, pois P > 0,05, não sendo significativo. Dessa forma, temos homogeneidade entre as variâncias entre os grupos. Mais um pressuposto verificado e podemos seguir com ANOVA de um fator. O oposto a homocedasticidade é a heterocedasticidade, ou seja, há heterogeneidade entre as variâncias dos grupos, quando o p-valor se mostra significativo.

Existem pelo menos dois meios básico de realizar a ANOVA no R. Um é usando a função aov() e outro usando a anova(). Na função anova() teremos que insertar dentro dela um modelo de regressão linear obrigatoriamente (podemos usar a função lm() para isso), e essa é a grande diferença entre elas. Entretanto, os resultados serão exatamente os mesmos.

Há um quadro típico que reúne os passos dos cálculos matemáticos até chegarmos ao final que é o valor de F calculado. Não iremos aprofundar muito em cada fórmula, pois não é o intuito do livro abordar os pormenores das contas à mão, mas sim como o R pode fazer isso por nós e sabermos interpretar seus resultados. Porém, o Quadro 13.1 é interessante para verificarmos o que o R retornará de resultados da ANOVA.

Quadro 13.1. Passos para os cálculos da ANOVA.

Fonte de variação	Soma dos Quadrados	Graus de Liberdade	Quadrados médios	F
Entre os grupos	$SQF = \sum_{i=1}^{k} n_i(\bar{x}_i - \bar{x})^2$	gl_F = k - 1	$QMF = \frac{SQF}{k-1}$	$F = \frac{QMF}{QMR}$
Dentro dos grupos	$SQR = \sum_{i=1}^{k}\sum_{j=1}^{n}(x_{ij} - \bar{x}_i)^2$	gl_R = n - k	$QMR = \frac{SQR}{n-k}$	

SQF = soma dos quadrados do fator; SQR = soma dos quadrados residuais; QMF = quadrado médio do fator; QMR = quadrado médio residual.

Como mostra o Quadro 13.1, primeiramente devemos calcular as somas de quadrados, tanto para o fator quanto para os resíduos (que é a variação dentro dos grupos). Com os graus de liberdade de cada fonte de variação, podemos calcular cada quadrado médio. Por fim, o F calculado é gerado pela divisão entre o quadrado médio do fator e o quadrado médio residual. A própria função relaciona esse valor ao F tabelado e calculará para gente o p-valor dessa análise.

Vamos rodar no R o nosso comando para o teste de ANOVA. As hipóteses de teste em nosso exemplo são:

H_0: não há diferença entre as médias de glicose entre os grupos.

H_1: há pelo menos uma média diferente entre os grupos.

```
result_anova <- aov(glicose ~ grupo, data = df7)
summary(result_anova)
             Df Sum Sq Mean Sq F value Pr(>F)
grupo         2 108505   54252   499.4 <2e-16 ***
Residuals   147  15969     109
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Usando agora a outra alternativa:

```
anova(lm(glicose ~ grupo, data = df7))
Analysis of Variance Table

Response: glicose
           Df Sum Sq Mean Sq F value    Pr(>F)
grupo       2 108505   54252  499.42 < 2.2e-16 ***
Residuals 147  15969     109
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Perceba que o R retorna todos os resultados do Quadro13.1. Em "Df" temos os graus de liberdade, em "Sum Sq" é retornado as respectivas somas de quadrados, em "Mean Sq" é retornado os respectivos quadrados médios, e em "F value" está o valor de F calculado. O p-valor que nos interessa está presente a coluna de resposta "Pr(>F)", que inclusive foi significativo (P < 0,001). Sendo assim, rejeitamos H_0. Há pelo menos uma média diferente de alguma outra. Mas qual ou quais?

Sem exageros, existem dezenas de métodos post-hoc, que são os testes que avaliam médias par a par. Alguns são mais conservadores (são mais difíceis de achar diferenças estatísticas) e outros são mais liberais nesse sentido (encontram diferenças entre médias mais facilmente). Os métodos considerados conservadores podem reduzir a chance de cometer um erro do tipo I (rejeitar H_0 quando é verdadeira) em comparações múltiplas, mas podem reduzir o poder estatístico do teste. Então, a ideia básica é testar alguns e ver qual se adequa melhor a responder a sua pergunta biológica.

Por hora, saiba que se houver homocedasticidade, devemos usar alguns testes *post hoc* como: Tukey, Bonferroni, Sidak, SNK, Duncan ou Scheffe. Existe ainda o de Dunnett, que é específico para comparar um grupo controle com os demais grupos. Quando não há homocedasticidade, podemos usar o teste de comparação de Tamhane, T3 ou C de Dunnett, Games-Howell.

Dentre todos os disponíveis, os mais usados são os testes de **Tukey** e o de **Bonferroni**. O primeiro mais liberal e o segundo mais conservador. Para o primeiro, iremos utilizar a função TukeyHSD(); basta usar o resultado salvo da ANOVA nessa função. Para a correção de Bonferroni, iremos utilizar a função pairwise.t.test() e especificaremos o parâmetro "p.adjust.method" para "bonferroni".

```
tukey <- TukeyHSD(result_anova)
tukey

  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = glicose ~ grupo, data = df7, var.equal = T)

$grupo
       diff       lwr       upr    p adj
B-A 61.2662  56.33071 66.201694 0.00e+00
C-A 51.6088  46.67331 56.544294 0.00e+00
C-B -9.6574 -14.59289 -4.721906 2.34e-05
```

Perceba que a função TukeyHSD() usa os resultados da ANOVA para calcular as diferenças entre as médias. O que nos mais importa é avaliar os valores de P ajustados: "p adj". Quando

significativo, menores ou iguais a 0,05, podemos dizer que naquela comparação as médias são estatisticamente diferentes. A primeira comparação demonstrada pelo R é entre as médias de B e A. O p-valor foi muito próximo de zero, ou seja, P < 0,001, portanto, significativo. Dessa forma, as medias de glicose entre os pacientes com diabetes controlada e não controlada são estatisticamente diferentes. A segunda comparação é entre C e A, que também foi significativa, perceba. Portanto, as médias de glicose entre pacientes que não sabiam do seu diagnóstico e pacientes com diabetes controlada também são diferentes. Já as médias entre C e B também houve diferença estatística.

Para visualizar essas diferenças, podemos rodar um gráfico usando os resultados gerados no teste Tukey. Veja o comando a seguir e seu resultado na Figura 13.5.

```
#Figura 13.5
plot(tukey)
```

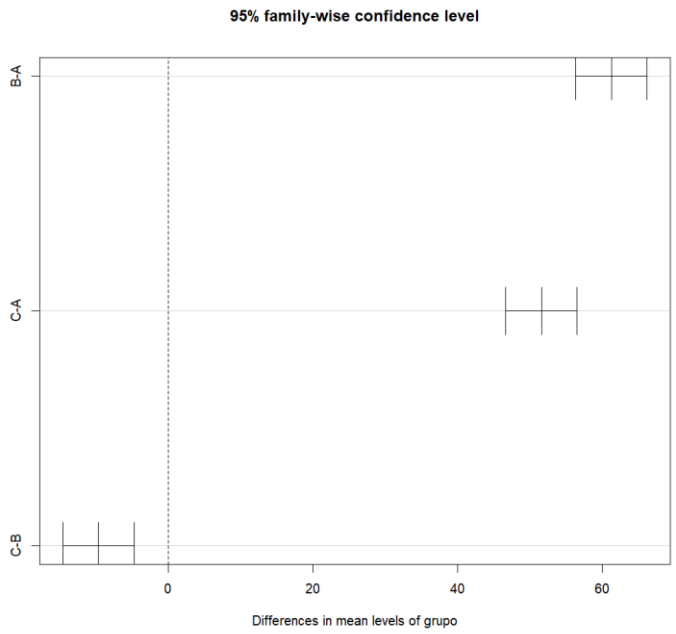


Figura 13.5. Comparação des médias par a par através do teste Tukey.

Vamos agora rodando o teste de Bonferroni e comparar seu resultado com o Tukey:

```
bonferroni <- pairwise.t.test(df7$glicose,
                              df7$grupo,
                              p.adjust.method = "bonferroni")
bonferroni$p.value
             A            B
B 6.054345e-63           NA
C 7.944176e-54 2.363364e-05
```

Vemos que o modo que essa função retorna os resultados de comparação é diferente. Contudo, ainda de simples interpretação. Ele retorna em forma de matriz e as comparações são feitas linha com coluna. Por exemplo, a comparação B-A obteve p-valor extremamente baixo, nos indicando diferença entre médias de glicose entre os grupos B e A. A comparação também foi significativa entre C-A e também entre C-B. E a comparação B e B retorna "NA", porque não faz sentido esse tipo de comparação. Em resumo, os números mudaram um pouco, mas chegamos ao mesmo resultado interpretativo. Nem sempre isso ocorre. Testar é uma das chaves e para testar outros testes post-hoc sugiro o pacote DescTools usando a função PostHocTest(). Dê uma olhada em sua documentação.

Para finalizar nosso entendimento da comparação entre médias, por que não visualizar essa comparação? Veja uma boa forma de apresentar esses resultados em um gráfico, na Figura 13.6. O p-valor do teste F da ANOVA não entrou na imagem porque seria redundante mostrar diferenças de médias e dizer que houve diferença entre médias.

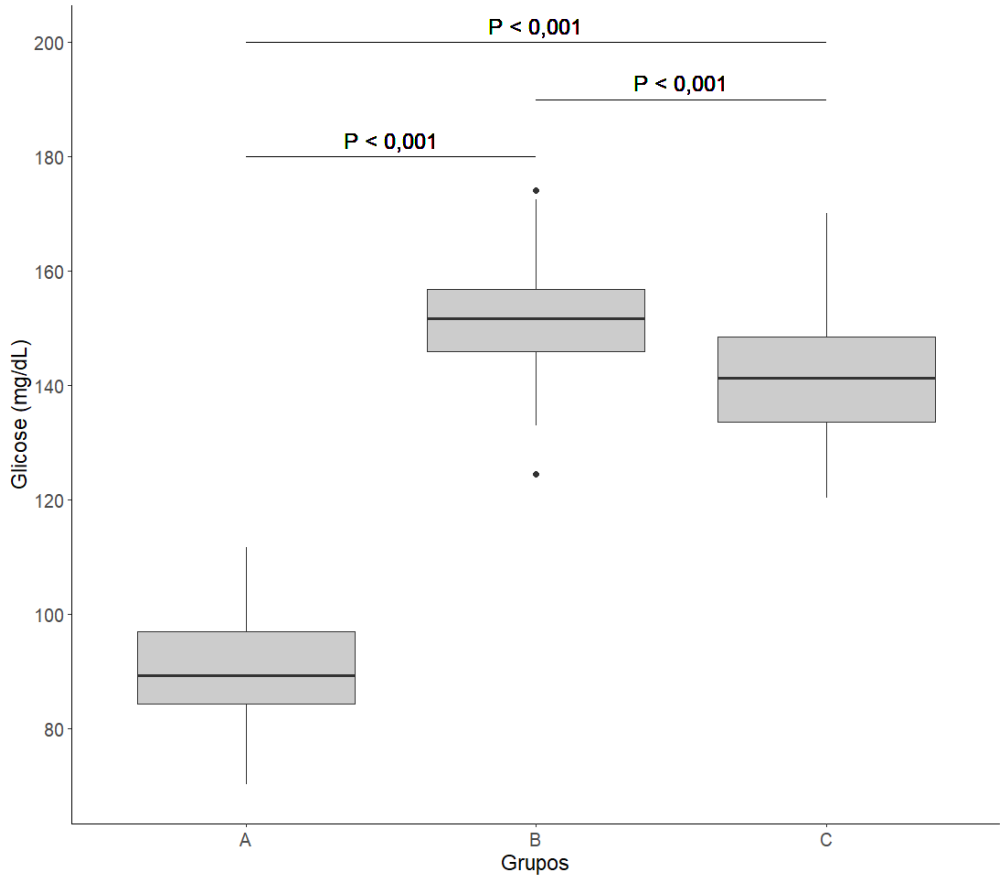


Figura 13.6. Comparação da média de glicose entre os grupos de estudo.

13.5.2 Teste Kruskal-Wallis (amostras independentes)

O teste Kruskal-Wallis, ou teste H, é um teste não paramétrico que testar a igualdade das medianas de três ou mais populações de forma independente. Ou seja, quando o pressuposto de normalidade ou homocedasticidade na comparação de três ou mais médias não forem atendidos, recorremos a ele. Teremos, portanto, a seguintes hipóteses estatísticas a testar:

H_0: As medianas das populações são iguais.

H_1: As medianas das populações são diferentes.

Interessantemente, o valor crítico do teste H é avaliado unilateralmente à direita. Diferente da grande maioria dos testes até agora abordados. Esse teste se baseia em pontos de amostras aleatórias, o que aproxima ele de uma distribuição de qui-quadrado, por isso, é essencial que cada grupo tenha pelo menos 5 observações.

Vamos pegar a essência da discussão do banco "df7" e reconstruí-lo mudando as distribuições das variáveis numéricas. Chamaremos esse novo banco de "df8".

```
set.seed(1234)
df8 <- data.frame(
  grupo = rep(c("A", "B", "C"), each = 50),
  glicose = round(c(A = rnorm(50,90,10),# glicose do grupo A
                    B = runif(50,130,200), # glicose do grupo B
                    C = rnorm(50,150,20) # glicose do grupo C
  ),2)
)

#ou importe os dados
library(rio)
df8 <- import('pasta_de_arquivos/df8.txt')

head(df8)
   grupo glicose
A1     A   77.93
A2     A   92.77
A3     A  100.84
A4     A   66.54
A5     A   94.29
A6     A   95.06
```

A interpretação das classes será a mesma: A = pacientes com diabetes controlada; B = pacientes com diabetes não controlada; C = paciente diabético que ainda não sabe o diagnóstico.

Com o banco montado ou importado para o R, vamos aos testes de normalidade.

```
shapiro.test(subset(df8, grupo == "A")$glicose)
      Shapiro-Wilk normality test
```

```
data:  subset(df8, grupo == "A")$glicose
W = 0.94504, p-value = 0.02142


shapiro.test(subset(df8, grupo == "B")$glicose)
      Shapiro-Wilk normality test

data:  subset(df8, grupo == "B")$glicose
W = 0.93128, p-value = 0.006181


shapiro.test(subset(df8, grupo == "C")$glicose)
      Shapiro-Wilk normality test

data:  subset(df8, grupo == "C")$glicose
W = 0.96662, p-value = 0.168
```

Como resultado, percebemos que as distribuições de glicose dos grupos A e B estão fora da normal. Seguiremos então com o teste de Kruskal-Wallis. Para isso, usaremos a função kruskal.test(). Usaremos a fórmula como método de entrada de dados para que ela entenda quais variáveis é a dependente e a independente.

```
kruskal.test(glicose ~ grupo, data = df8)
      Kruskal-Wallis rank sum test

data:  glicose by grupo
Kruskal-Wallis chi-squared = 104.64, df = 2, p-value < 2.2e-16
```

Analisando o valor de P do teste, percebemos que é muito pequeno, inferior a 0,05. Portanto, é significativo e devemos rejeitar H_0. Dessa forma, há diferenças entre medianas. Só não sabemos ainda de qual com qual.

Para revelarmos onde há diferença de medianas, iremos utilizar a função pairwise.wilcox.test(), muito similar à usada após a ANOVA. Essa função em específico faz múltiplas comparações utilizando o teste de Mann-Whitney como base. Contudo, podemos criar ajustes nela, modificando o parâmetro "p.adjust.method". Vamos exemplificar um ajuste utilizando Bonferroni.

```
pairwise.wilcox.test(df8$glicose, df8$grupo,
                     p.adjust.method = "bonferroni")
      Pairwise comparisons using Wilcoxon rank sum test with continuity correction

data:  df8$glicose and df8$grupo

  A      B
B <2e-16 -
C <2e-16 0.0017

P value adjustment method: bonferroni
```

Como resultado, vemos que todos os p-valores são significativos. As comparações entre A-B e A-C foram ambas P < 0,001 e a comparação entre B-C obteve P = 0,0017. Logo, todas as medianas são estatisticamente diferentes entre si. Vamos solicitar ao R que nos retorne os valores específicos de mediana para cada grupo e vamos gerar um *boxplot* para analisarmos isso visualmente (Figura 13.7).

```
median(df8[df8$grupo == 'A',2])
[1] 84.645

median(df8[df8$grupo == 'B',2])
[1] 164.99

median(df8[df8$grupo == 'C',2])
[1] 149.215
```

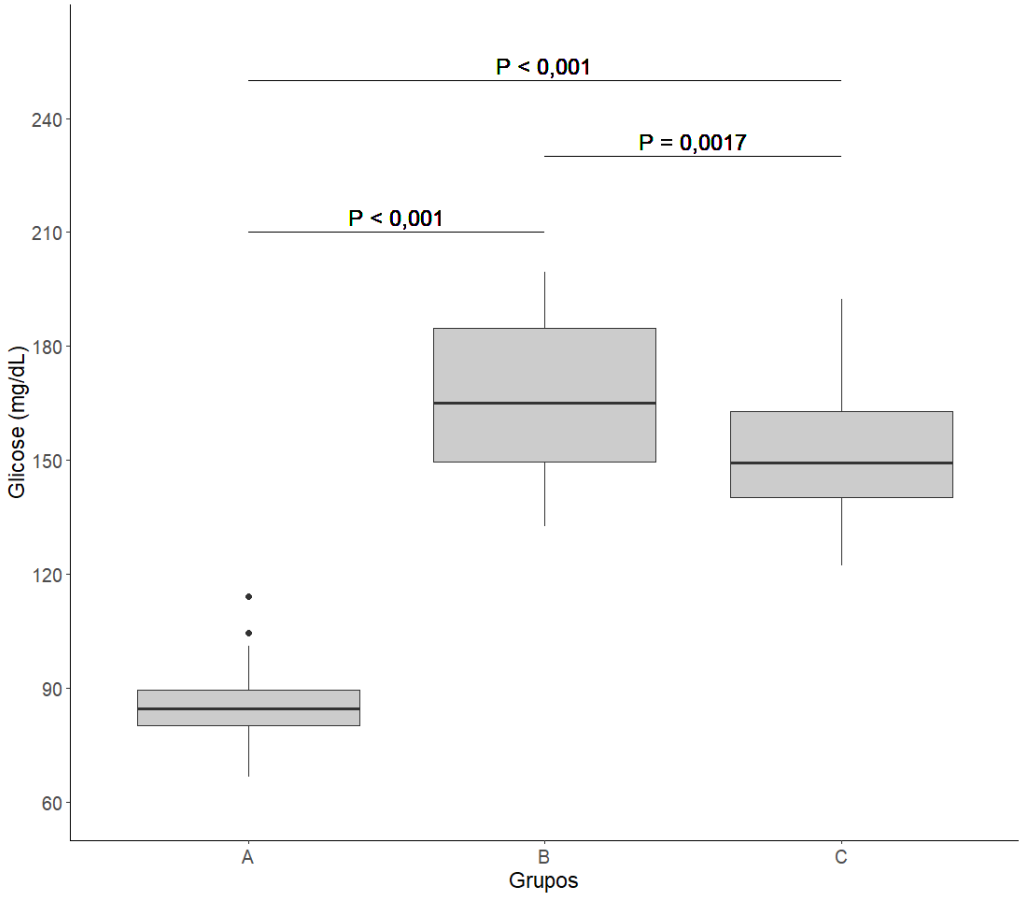


Figura 13.7. *Boxplot* mostrando a taxa de glicose entre grupos de diabéticos.
A = diabetes controlada; B = diabetes não controlada; C= paciente diabético, mas sem saber diagnóstico.

Chegamos à conclusão que para essa amostragem, o grupo com a diabetes controlada tem taxas medianas menores, enquanto que o grupo que tem diabetes não controlada tem taxas medianas maiores entre os três grupos, tendo o grupo C taxa intermediárias de glicose sérica.

13.6 ANOVA DE MEDIDAS REPETIDAS (AMOSTRAS PAREADAS)

Se no teste t pareado nós analisamos as médias dos mesmos indivíduos em dois momentos ou condições diferentes, agora podemos analisar suas médias em três ou mais momentos ou condições diferentes. Essa é a ANOVA de medidas repetidas, um teste pareado. Um exemplo muito comum é quando pesquisadores desejam testar uma nova terapia e avaliar antes, após e muito após ao tratamento (tempo zero, após 3 dias, após 15 dias, após 1 mês, após 1 anos).

Para exemplificar, vamos supor que um imunologista deseje avaliar a titulação (quantificação) de anticorpos do tipo IgG após aplicação de uma nova vacina contra raiva, em camundongos, e assim saber se sua fórmula é eficaz ou não (produz imunidade adequada ou não). Muitas vezes os anticorpos são dosados usando uma técnica chamada de ELISA (do inglês *Enzyme-Linked Immunosorbent Assay*). Os resultados das dosagens podem sair pela absorbância: quanto maior o valor, mais foi detectado na amostra a quantidade de anticorpo.

Ele seguirá o seguinte passo a passo:

1. Dosar IgG específica no tempo zero, sem tomar vacina;
2. Vai aplicar a vacina;
3. Dosar IgG específica um dia depois da vacina;
4. Dosar IgG específica depois de 15 dias;
5. Vai realizar um desafio 15 dias depois (dia 30 após a administração da vacina), ou seja, administrar antígeno do vírus da raiva para verificar a resposta imune do animal;
6. Dosar IgG específica 1 dia depois do desafio.

Resumindo o problema, o pesquisador terá quatro conjunto de dados numéricos da dosagem de IgG sérica: (1) dosagem antes da vacina; (2) dosagem um dia depois da vacina; (3) dosagem 15 dias depois da vacina; (4) dosagem 31 dias depois da vacina.

Existem alguns pressupostos antes de correr a ANOVA de medidas repetidas:

- A variável dependente deve ser contínua;
- As distribuições da variável contínua em cada tempo devem estar na normal;
- Os dados devem ser aleatórios, ou seja, livre de vieses tendencioso;
- Esfericidade das classes.

A esfericidade é parecida com a homocedasticidade que é usada na ANOVA de uma via e nas regressões lineares. O que é desejável é que as variâncias nas classes sejam estatisticamente

iguais. Caso a gente perca esse pressuposto, os resultados deixam de ter a confiabilidade que desejamos. Para avaliar a esfericidade, usamos o teste de Mauchly.

Na realidade, usando dados reais é mais fácil encontrar variâncias estatisticamente diferentes, pois depende do número de tempos ou classes, depende do número amostral, depende da natureza dos dados, enfim. Para lidar com análises cuja variâncias sejam diferentes, precisamos aplicar uma correção ao p-valor gerado pela ANOVA de medidas repetidas. Vamos aprender como se usa tais correções, porque é muito provável que você venha a utilizar um dia. Podemos utilizar a correção de Greenhouse-Geisser ou de a Huynh-Feldt.

Nesse tipo de análise, a melhor forma de construir nosso banco de dados para os testes que virão é ainda uma coluna para cada variável. Devemos ter, portanto, o momento da coleta e a dosagem de IgG. Temos duas variáveis então, uma categórica e outra numérica. Chamaremos esse nosso novo banco de "df9". A variável "id" é a identificação de cada indivíduo; aqui colocamos a sequência de 1 a 25 para identifica-los, mas é uma variável classificadora. A variável "tempos" indica os quatro tempos diferentes da seguinte maneira: T0, T1, T2 e T3. Por fim, a variável "igg" é a dosagem sérica de anticorpo detectável.

```
set.seed(777)
df9 <- data.frame(
  id = c(rep(seq(1,50),4)),
  tempos = c(rep('T0',50),rep('T1',50),rep('T2',50),rep('T3',50)),
  igg = c(abs(rnorm(50,1,0.5)),abs(rnorm(100,4,3)),abs(rnorm(50,8,3)))
)

#ou importe o banco de dados
library(rio)
df9 <- import('pasta_de_arquivos/df9.txt')
```

Como as variáveis "id" e "tempos" não são numéricas, para essa análise precisaremos transformá-las em fator. Usando head() ou str() podemos checar os dados.

```
df9$id <- as.factor(df9$id)
df9$tempos <- as.factor(df9$tempos)

head(df9)
  id tempos       igg
1  1     T0 1.2448931
2  2     T0 0.8007293
3  3     T0 1.2554182
4  4     T0 0.8005940
5  5     T0 1.8193430
6  6     T0 1.3106370

str(df9)
'data.frame':      200 obs. of  3 variables:
```

```
 $ id    : Factor w/ 50 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 ...
 $ tempos: Factor w/ 4 levels "T0","T1","T2",..: 1 1 1 1 1 1 1 1 ...
 $ igg   : num  1.245 0.801 1.255 0.801 ...
```

Perceba que a função str() verifica a estrutura do objeto. Ela é interessante para verificarmos as classes das variáveis de forma genérica.

Vamos testar a normalidade das distribuições em cada tempo:

```
shapiro.test(subset(df9, tempos == "T0")$igg)
      Shapiro-Wilk normality test

data:  subset(df9, tempos == "T0")$igg
W = 0.98509, p-value = 0.7753


shapiro.test(subset(df9, tempos == "T1")$igg)
      Shapiro-Wilk normality test

data:  subset(df9, tempos == "T1")$igg
W = 0.9695, p-value = 0.2209


shapiro.test(subset(df9, tempos == "T2")$igg)
      Shapiro-Wilk normality test

data:  subset(df9, tempos == "T2")$igg
W = 0.97267, p-value = 0.2963


shapiro.test(subset(df9, tempos == 'T3')$igg)
      Shapiro-Wilk normality test

data:  subset(df9, tempos == "T3")$igg
W = 0.98394, p-value = 0.7251
```

Pelos resultados, todos os p-valores não foram significativos, o que nos indica que as distribuições estão na normal. Pressupostos atendidos, usaremos ANOVA. Para isso, recorreremos à função ezANOVA() do pacote ez. Esse pacote tem funções que fazem diversos testes estatísticos que já comentamos ao longo do livro. Quando tiver tempo, sugiro dar uma olhada na sua documentação e testar demais funções. Vamos instalá-lo e abrir sua biblioteca.

```
install.packages('ez')
library(ez)
```

Precisamos preencher alguns parâmetros obrigatórios da função ezANOVA(). O primeiro deles é o parâmetro “data”, onde indicaremos o banco de dados. Em “dv” especificaremos a variável dependente, que é igg. Em “wid” indicaremos a identificação dos indivíduos com a nossa variável id. Em “within” especificaremos a variável categórica que define os diferentes tempos. O

parâmetro "detailed" colocamos igual a TRUE para que o R retorne detalhes do modelo de ANOVA construído. Por fim, especificamos o parâmetro "type" igual a 3 para que a função calcule a ANOVA igualmente como outros programas estatísticos como o SPSS e o SAS, assim, nós teremos repetibilidade de resultados.

Vamos rodar o comando salvando os resultados em um objeto que chamaremos de "anova_rep".

```
anova_rep <- ezANOVA(data = df9,
                     dv = igg,
                     wid = id,
                     within = tempos,
                     detailed = TRUE,
                     type = 3)

anova_rep

$ANOVA
       Effect DFn DFd     SSn    SSd       F    p p<.05     ges
1 (Intercept)   1  49 3637.01 434.97 409.708 1.9067e-25     * 0.75212
2      tempos   3 147 1056.77 763.63  67.809 1.3849e-27     * 0.46855

$`Mauchly's Test for Sphericity`
  Effect         W          p p<.05
2 tempos 0.7664448 0.02645508     *

$`Sphericity Corrections`
  Effect     GGe      p[GG] p[GG]<.05     HFe      p[HF] p[HF]<.05
2 tempos 0.87196 2.2604e-24         * 0.92556 1.0205e-25         *
```

Analisando os resultados gerados, o primeiro que o R retorna é o teste F, que é específico para a ANOVA. Essa hipótese H_0 em teste é se as médias nos diferentes tempos são iguais entre si. O p-valor se encontra na coluna de resultados "p p<.05". Ele foi muito significativo (P<0,001), contudo, ainda não devemos toma-lo como verdade, pois precisamos avaliar ainda a esfericidade pelo teste de Mauchly, que está logo abaixo nos resultados.

No teste de Mauchly estamos testando as hipóteses:

H_0: as variâncias são iguais.

H_1: há variâncias diferentes.

Pelos resultados, temos um p-valor significativo (P = 0,026). Sendo assim, rejeitamos H_0 para o teste de Mauchly, há variância estatisticamente diferentes. Por conta disso, não devemos aceitar ou rejeitar as hipóteses do teste F nesse caso. Para as situações onde não há esfericidade (variâncias estatisticamente diferentes), devemos avaliar os p-valores das correções de

esfericidade: Greenhouse-Geisser (GG) e Huynh-Feldt (HF). Ambas são mais conservadoras do que a ANOVA, mas a correção GG ainda é mais conservadora do que HF.

Pelos resultados que o R gerou, o p-valor para GG (p[GG]) e para HF (p[HF]) foram significativos. Então, definitivamente concluímos que há diferença da quantidade de anticorpos séricos ao longo dos tempos de estudo.

Contudo, ainda não sabemos onde estão essas diferenças. Para isso, vamos usar a função pairwise.t.test() que calculará testes t pareados par a par entre os diferentes tempos. Para isso, é importantíssimo especificar o parâmetro "paired = TRUE". Usaremos a correção de Bonferroni também, mas essa não é obrigatória, mas deixará nosso teste mais conservador.

```
pairwise.t.test(df9$igg, df9$tempos,
                paired = TRUE,
                p.adjust.method = 'bonferroni')
      Pairwise comparisons using paired t tests

data:  df9$igg and df9$tempos

   T0      T1      T2
T1 2.0e-10 -       -
T2 8.7e-11 1       -
T3 < 2e-16 8.0e-09 9.6e-06

P value adjustment method: bonferroni
```

Tomando como base a matriz que o R retornou das múltiplas comparações pareadas, perceba que todos os tempos são diferentes entre si ($P \leq 0,05$), exceto a comparação entre os tempos T1 e T2 ($P = 1,00$). Para enxergarmos melhor essas diferenças, vamos fazer um gráfico do tipo *boxplot*, que está presente na Figura 13.8.

```
#Figura 13.8
boxplot(igg ~ tempos, data = df9,
        ylab = 'Dosagem de IgG em nm',
        xlab = 'Tempos')
```

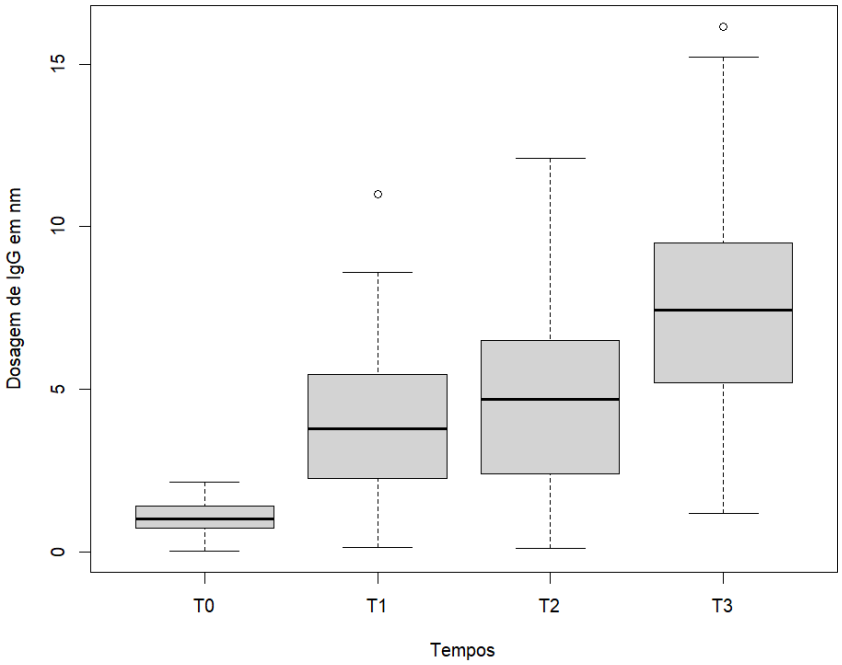


Figura 13.8. Distribuição das dosagens de IgG entre os tempos T0, T1, T2 e T3.

Podemos ver na Figura 13.8 que a dosagem de IgG antes da vacina são bem próximas de zero (T0), então, em média, é o tempo com dosagem mais baixa. Na análise laboratorial, as vezes é difícil que um teste dê zero mesmo quando não há presença da molécula de interesse. Esse fato ocorre por conta dos corantes, dos reagentes, de outras moléculas que podem gerar reações cruzadas, da própria calibragem da máquina, enfim. Apesar do valor baixo, existirá sempre uma leitura do detector do aparelho. O pesquisador deverá se preocupar com isso antes de fechar qualquer conclusão, estabelecendo ponto de corte mínimo, por exemplo. É uma questão *a priori* extra-estatística.

Já as dosagens entre o dia após a vacina e depois de 15 dias são bem semelhantes, pois as caixas estão no mesmo patamar, porém maiores do que em T0; lembrar que T1 e T2 têm médias estatisticamente iguais. Por fim, a dosagem após o desafio (T3) aparece como a média mais elevada e também é diferente das demais.

Podemos representar visualmente melhor esses resultados, colocando os valores de P ou os asteriscos relacionados à significância entre os boxplot, como na Figura 13.9.

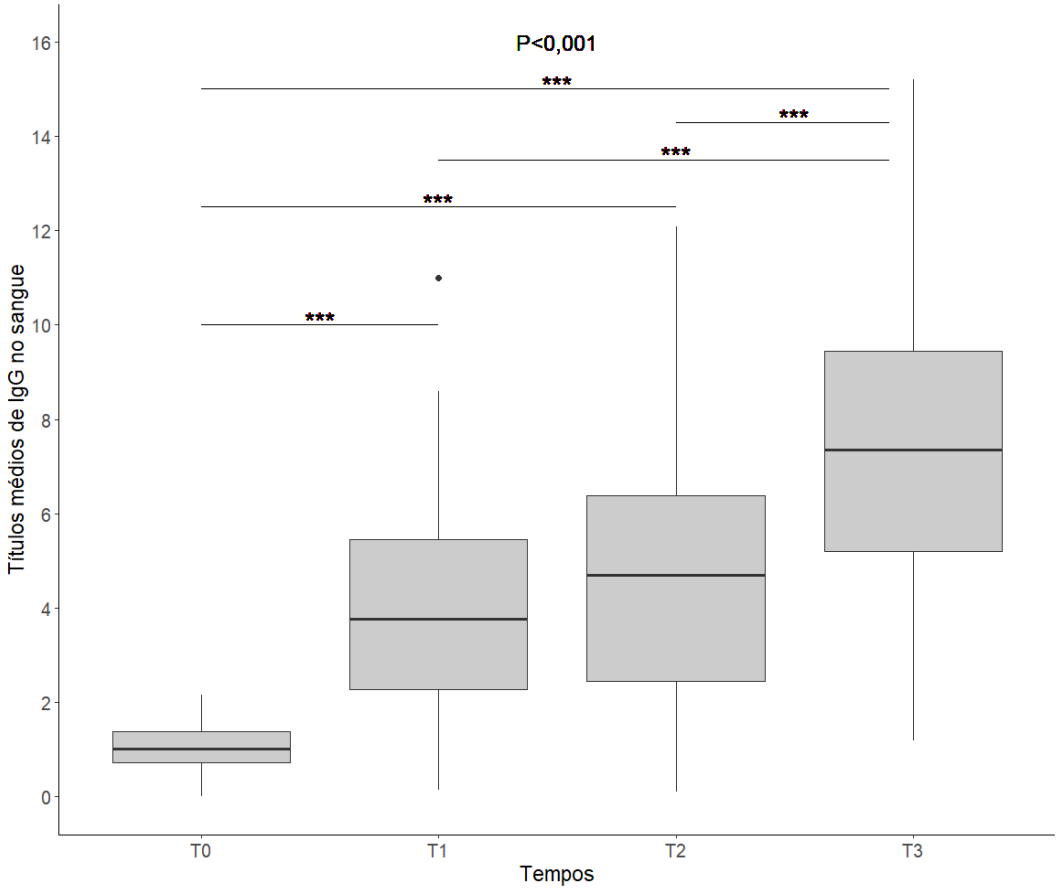


Figura 13.9. Comparação entre as dosagens médias de IgG raiva-específica antes e após vacina.

13.7 TESTE DE FRIEDMAN (TESTE NÃO PARAMÉTRICO PARA AMOSTRAS PAREADAS)

O teste de Friedman é uma alternativa não paramétrica do teste de ANOVA para medidas repetidas. Como outros testes não paramétricos no uso de amostras numéricas, o Friedman não analisa médias, mas sim medianas. Tecnicamente ele é similar ao Kruskal-Wallis por considerar postos das distribuições dos valores numéricos e não as médias.

Então, as hipóteses de teste serão:

H_0: as medianas são iguais em todas as k populações.

H_1: há pelo menos uma diferença entre as medianas das k populações.

Utilizaremos o mesmo exemplo do capítulo anterior, mas mudaremos a seleção dos dados aleatórios. Chamaremos esse novo bando de dados de “df10”.

```
set.seed(444)
df10 <- data.frame(
```

```
  id = c(rep(seq(1,50),4)),
  tempos = c(rep('T0',50),rep('T1',50),rep('T2',50),rep('T3',50)),
  igg = c(abs(rnorm(50,1,0.5)),abs(rnorm(100,4,3)),abs(rnorm(50,8,3)))
)

#ou importe o banco de dados
library(rio)
df10 <- import('pasta_de_arquivos/df10.txt')

#transformando em fatores as variáveis categóricas
df10$id <- as.factor(df10$id)
df10$tempos <- as.factor(df10$tempos)

head(df10)
  id tempos       igg
1  1     T0 0.5190563
2  2     T0 0.8599314
3  3     T0 0.1720622
4  4     T0 1.0747721
5  5     T0 1.5162066
6  6     T0 0.7439742
```

Vamos testar a normalidade dos dados em cada em dos tempos do estudo:

```
shapiro.test(subset(df10, tempos == "T0")$igg)
       Shapiro-Wilk normality test

data:  subset(df10, tempos == "T0")$igg
W = 0.9684, p-value = 0.1992


shapiro.test(subset(df10, tempos == "T1")$igg)
       Shapiro-Wilk normality test

data:  subset(df10, tempos == "T1")$igg
W = 0.91527, p-value = 0.001598


shapiro.test(subset(df10, tempos == "T2")$igg)
       Shapiro-Wilk normality test

data:  subset(df10, tempos == "T2")$igg
W = 0.92853, p-value = 0.004863


shapiro.test(subset(df10, tempos == 'T3')$igg)
       Shapiro-Wilk normality test

data:  subset(df10, tempos == "T3")$igg
W = 0.98203, p-value = 0.6404
```

Pelos resultados de normalidade retornados, as distribuições nos tempos de estudo T0 e T3 estão na normal. Entretanto, em T1 e T2 as distribuições estão fora da normal. É o pressuposto

necessário para descartamos ANOVA de medidas repetidas e compararmos os tempos usando teste não paramétrico de Friedman.

Podemos usar a função friedman.test() própria do R base para rodar os cálculos necessários. Podemos escrever a sintaxe utilizando uma fórmula, contudo, ela é sensivelmente diferente. Precisamos primeiramente colocar a variável dependente (numérica), o til que a separará, a variável independente que difere os tempos, barra vertical e a variável de identificação dos indivíduos ou observações (que no nosso exemplo identificamos como "id").

```
friedman.test(igg ~ tempos | id, data = df10)

      Friedman rank sum test

data:  igg and tempos and id
Friedman chi-squared = 84.984, df = 3, p-value < 2.2e-16
```

Percebemos pelo p-valor do resultado da análise de Friedman que devemos rejeitar H_0, pois o p-valor foi significativo (P<0,001). Portanto, há medianas diferentes entre si entre os tempos de estudo. Mas ainda não sabemos quais são essas.

Para isso, precisamos correr pelo menos um teste post-hoc. O mais comum é fazermos o teste Wincoxon para amostras pareadas, par a par. Lembrando que já trabalhamos com esse teste nesse capítulo. Entretanto, a função wilcox.test() do R base que usamos anteriormente só compara populações duas de cada vez. Como o nosso exemplo temos quatro populações a comparar uma com a outra (são seis comparações no total), podemos fazer isso facilmente usando a função wilcox_test() do pacote rstatix. Ela retornará os testes de comparação par a par diretamente e cada comparação é posta linha a linha. Caso você ainda não tenha esse pacote, instale-o. Usaremos a correção de Bonferroni como exemplo para deixar nossas comparações de medianas mais conservadores, ou seja, um pouco mais difícil de acharmos diferenças significativas (mas isso não é obrigado e basta omitir o parâmetro "p.adjust.method" caso não queira).

```
install.packages('rstatix')
library(rstatix)

wilcox_test(igg ~ tempos,
            data = df10,
            paired = TRUE,
            p.adjust.method = 'bonferroni')

# A tibble: 6 × 9
 .y. group1  group2    n1    n2 statistic       p         p.adj p.adj.signif
*<chr> <chr>  <chr>  <int> <int>     <dbl>    <dbl>         <dbl> <chr>
1 igg   T0     T1       50    50        43 9.81e- 9 0.0000000589  ****
2 igg   T0     T2       50    50        11 1.51e- 9 0.00000000906 ****
```

```
3 igg   T0     T3       50    50         1 8.28e-10 0.00000000497 ****
4 igg   T1     T2       50    50       566 4.93e- 1 1             ns
5 igg   T1     T3       50    50       182 1.12e- 5 0.0000672     ****
6 igg   T2     T3       50    50       195 1.98e- 5 0.000119      ***
```

Perceba que a função retorna os valores de P (em p) e os valores de P ajustados (em p.adj) por Bonferroni. Para facilitar a nossa vida a função retorna a significância no padrão de asteriscos (em p.adj.signif).

Pelos resultados, observe que todas as comparações foram estatisticamente significativas, exceto a comparação entre o T1 e T2 (P = 0,493; $P_{ajustado}$ = 1,00).

Existem alguns testes post-hoc que são específicos para essa análise de Friedman, sendo uma outra alternativa a se analisar, como o teste de Conover, teste exato de Eisinga, teste de Miller, teste de Nemenyi e teste de Siegel e Castellan. Podemos utilizar funções do pacote PMCMRplus, que são respectivamente: frdAllPairsConoverTest(), frdAllPairsExactTest(), frdAllPairsMiller-Test(), frdAllPairsNemenyiTest() e frdAllPairsSiegelTest(). A sintaxe é a mesma para todas elas. Devemos pôr a variável dependente primeiro, depois a independente e logo após a variável de identificação de cada observação (em nosso exemplo: id). Vamos usar novamente a correção de Bonferroni para ilustrar um ajuste. Iremos exemplificar aqui com a função de Siegel e Castellan, pois outros softwares estatísticos costumam usar esse método.

```
install.packages('PMCMRplus')
library(PMCMRplus)

frdAllPairsSiegelTest(df10$igg,
                      df10$tempos,
                      df10$id,
                      p.adjust.method = 'bonferroni')
      Pairwise comparisons using Siegel-Castellan all-pairs test for a two-way bal-
anced complete block design

data: y, groups and blocks

   T0      T1     T2
T1 2.3e-07 -      -
T2 2.4e-08 1.0000 -
T3 < 2e-16 0.0022 0.0090

P value adjustment method: bonferroni
```

Perceba que os resultados retornados são dados em uma matriz de comparação e nela estão os seus p-valores ajustados, como solicitamos. Novamente, a única comparação que não foi significativa em nosso exemplo foi entre T1 e T2. Então, os títulos de IgG sanguíneos são iguais apenas entre os tempos T1 e T2 da pesquisa.

Vamos fechar esse entendimento avaliando o *boxplot* através da Figura 13.10, que tem intencionalmente o mesmo padrão das Figuras 13.8 e Figura 13.9.

```
#Figura 13.10
boxplot(igg ~ tempos, data = df10,
        ylab = 'Dosagem de IgG em nm',
        xlab = 'Tempos')
```

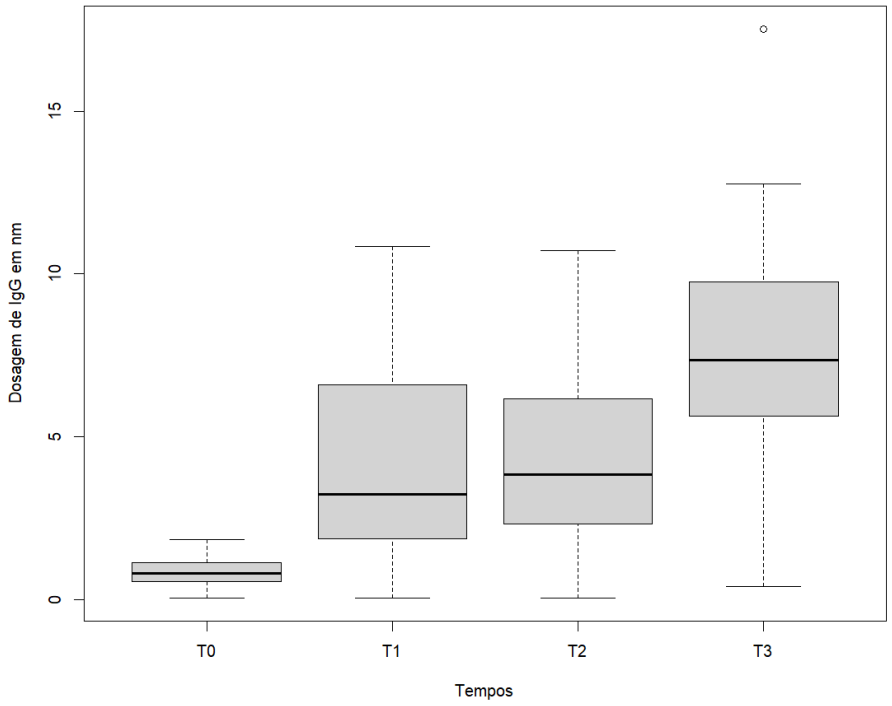


Figura 13.10. Distribuição das dosagens de IgG entre os tempos T0, T1, T2 e T3.

Visualmente falado, as caixas de medianas estatisticamente iguais estão no mesmo patamar (T1 e T2). Podemos dizer que no tempo T0 os níveis de IgG são basais, enquanto que em T1 e T2 os níveis são intermediários, contudo, maiores do que T0. Os maiores níveis de IgG ocorrem em T3, após o desafio com o antígeno.

Capítulo 11

Testes de Concordância

Em um grupo de estudo há dois médicos pneumologistas que ficam encarregados de diagnosticar asma em uma determinada população. A intensão do grupo é recrutar a maior quantidade de asmáticos possíveis no município. Mas, há uma questão intrínseca sobre a asma que você precisa saber: ela é complexa e multifatorial. O diagnóstico certeiro não é tão simples em algumas situações. Então, surge uma pergunta: será que esses dois médicos dão o mesmo diagnóstico da doença a todos os pacientes igualmente? Ou seja, será que eles concordam em todas as situações?

Ainda, vamos supor que esse mesmo grupo aplique um questionário de qualidade de vida para esses pacientes. Nesse questionário, algumas variáveis resultarão em um determinado escore, um número dentro de um espectro. Por exemplo, se esse escore for alto, pior a qualidade de vida do paciente; tendo escore baixo, estima-se que melhor é a sua qualidade de vida. Para essa tarefa foram selecionadas duas enfermeiras. Aqui há outra dúvida: será que elas aplicam esse questionário de forma similar a ponto de gerar exatamente os mesmos escores para cada pacientes? Será que elas concordam no preenchimento dessa e de outras variáveis numéricas?

Para resolver essa questão, temos testes estatísticos chamados de testes de concordância. Na primeira situação, perceba que a preocupação está em uma variável categórica: diagnosticar asma ou não asma. Para testar essa concordância podemos utilizar o kappa de Cohen para responder os primeiros questionamentos de diagnóstico. O questionamento sobre as enfermeiras está sobre uma variável numérica: o valor do escore de qualidade de vida. Para essa questão, usamos a estatística de Bland-Altman.

Esses testes validam a confiabilidade e a consistência de julgamentos entre os avaliadores, o que é muito importante na bioestatística e na epidemiologia, especialmente quando há a aplicação de questionários independente da sua finalidade.

14.1 KAPPA DE COHEN (CONCORDÂNCIA DE CARACTERÍSTICA)

Como dito acima, a estatística kappa nos ajuda a estipular o nível de concordância entre avaliadores com base nos seus dados categóricos coletados. Existem outras estatísticas para esse fim, mas o que aprenderemos aqui foi criada por Jacob Cohen, publicada em 1960. Matematicamente, o cálculo do coeficiente se baseia na concordância real observada e a concordância casual.

Seu coeficiente calculado (o próprio kappa) varia entre -1 a 1. Valores negativos indicam que há mais discordância do que concordância entre os entrevistadores; quanto mais próximo de -1, mais discordantes são eles. O valor -1 indica discordância completa. Dentre os valores positivos, a tabela 14.1 modificada de McHugh (2012) nos ajuda a interpretar o nível de concordância. Em suma, quanto mais próximo de +1, mais forte é a concordância entre eles. O valor +1 é concordância perfeita.

Tabela 14.1 Nível de concordância segundo valor de kappa.

Valor de kappa	Nível de concordância	% de dados confiáveis
0,00-0,20	Nenhum	0-4%
0,21-0,39	Mínimo	4-15%
0,40-0,59	Fraco	15-35%
0,60-0,79	Moderado	35-63%
0,80-0,90	Forte	64-81%
> 0,90	Quase perfeito	82-100%

Fonte: McHugh, 2012 (modificado).

Um exemplo do que podemos fazer com a estatística kappa de Cohen pode ser visto no trabalho de Cardoso de colaboradores (2015). Eles testaram a concordância entre os testes de detecção de citomegalovírus em crianças em amostras de urina e saliva. Obtiveram um valor de kappa de 0,887 (P<0,001), ou seja, uma forte concordância. Em suma, quando o citomegalovírus era detectado na urina, também o era na saliva na grande maioria das vezes. O mesmo podemos pensar quando a detecção era negativa: quando não estava presente na urina, também não se via traços na saliva na maioria das vezes.

No R, vamos praticar usando funções do pacote irr, o qual apresenta diversas estatísticas de concordância. Se você se interessar em se aprofundar nesse quesito, sugiro dar uma olhada com calma na sua documentação. O banco de dados que iremos trabalhar também está presente dentro desse pacote e se chama "diagnoses". Nele, há o julgamento diagnóstico de 30 pacientes por seis psiquiatras diferentes (rater1 a rater6). Os avaliadores diagnosticaram os pacientes entre distúrbio de personalidade, neurose, esquizofrenia, depressão e outros.

Para calcularmos o valor de kappa, usaremos a função kappa2(), que serve para avaliarmos a concordância entre dois avaliadores. Dentro da função, devemos determinar as colunas dos avaliadores. No exemplo abaixo, escolheremos primeiramente o avaliador 1 e 2.

```
install.packages('irr')
library(irr)

#abrindo banco de dados presente no pacote irr
data(diagnoses)

#verificando a disposição dos dados dos avaliadores 1 e 2
head(diagnoses[1:2])
                    rater1                  rater2
1             4. Neurosis             4. Neurosis
2 2. Personality Disorder 2. Personality Disorder
3 2. Personality Disorder        3. Schizophrenia
4                5. Other                5. Other
5 2. Personality Disorder 2. Personality Disorder
6           1. Depression           1. Depression

#calculando kappa para avaliadores 1 e 2
kappa2(diagnoses[,c(1,2)])

 Cohen's Kappa for 2 Raters (Weights: unweighted)

 Subjects = 30
   Raters = 2
    Kappa = 0.651

        z = 7
  p-value = 2.63e-12
```

A primeira coisa que você precisa perceber é como o banco de dados deve ser construído para que a função calcule o valor de kappa corretamente. Cada avaliador será uma coluna e cada paciente/observação será uma linha. Perceba também que as variáveis são categóricas. Cuidado quando você estiver trabalhando com variáveis codificadas numericamente, pois em algum momento pode ocorrer erros ou falsos resultados.

Como resultado, observamos que o valor de kappa foi 0,651. Uma concordância moderada, segundo a Tabela 14.1. Perceba que ele retorna um valor de P, que foi significativo. Se você visualizar o banco de dados, perceberá que eles estão concordando em 22 diagnósticos dentre os 30 pacientes em análise. Isso é 73,33% das observações. De fato, é uma porcentagem boa, mas nem tanto assim.

Você pode estar se perguntando: será que existem um meio de calcularmos um kappa de concordância entre todos os avaliadores? A resposta é sim. Usando o kappa de Fleiss, podemos tirar conclusões de concordância para mais de dois avaliadores. Além disso, podemos avaliar qual classe ou categoria os avaliadores estão concordando e discordando mais. Isso é muito bom no processo de padronização de entrevistas.

Vamos utilizar ainda o banco de diagnóstico entre os seis psiquiatras. Usaremos agora a função kappam.fleiss() e nela pôr o nome do banco como parâmetro inicial. Vamos adicionar o parâmetro "detail = TRUE" para que o R retorne os detalhes analíticos sobre cada categoria (cada diagnóstico).

```
kappam.fleiss(diagnoses, detail=TRUE)
 Fleiss' Kappa for m Raters

 Subjects = 30
   Raters = 6
    Kappa = 0.43

        z = 17.7
  p-value = 0

                         Kappa      z p.value
1. Depression            0.245  5.192   0.000
2. Personality Disorder  0.245  5.192   0.000
3. Schizophrenia         0.520 11.031   0.000
4. Neurosis              0.471  9.994   0.000
5. Other                 0.566 12.009   0.000
```

Primeiramente, percebemos que há um kappa deduzido (0,43), o que seria uma concordância fraca, mas com p-valor significativo (p-value = 0; ou seja, P<0,001).

Antes de mais nada, você precisa perceber criticamente que se nós aumentarmos a quantidade de avaliadores, é mais provável que eles discordem em algum ponto. Dessa forma, quanto mais avaliadores, esperamos que haja menos concordância. Mas quantos avaliados ter, então? Responder essa pergunta não é tão fácil em boa parte das situações, contudo, é importante que haja um avaliador experiente, para que ele seja uma referência para os demais.

Seguindo, no final dos resultados, podemos observar que os avaliadores estão discordando mais ao diagnosticar depressão e distúrbios de personalidade (kappa = 0,25; P<0,001). Enquanto que "Other" é onde eles mais têm concordado (kappa = 0,566; P<0,001), ou seja, concordam mais quando definem que não é nenhuma das quatro causas psiquiátricas (depressão, distúrbio de personalidade, esquizofrenia ou neurose).

14.2 BLAND-ALTMAN (CONCORDÂNCIA NUMÉRICA)

O método de Bland-Altman é diferente dos demais acima, pois com ele podemos avaliar a concordância entre valores numéricos, o que requer um pouco mais de atenção. Muitos pesquisadores equivocadamente têm utilizado a análise de correlação para esse tipo de avaliação, o que não

é adequado. Muitos se equivocam por não conhecer a diversidade de métodos estatísticos que existem disponíveis. Normalmente isso vem com o tempo e a experiência que os artigos científicos proporcionam.

Hirakata e Camey (2009) explicam que a correlação avalia a relação entre as variáveis, não a concordância. Se um aparelho de aferir pressão fornecesse os valores de pressão sistólica 80, 95, 96 e 100 mmHg de quatro indivíduos e um segundo aparelho fornecesse respectivamente para os mesmos indivíduos 90, 105, 106 e 110 mmHg, a correlação entre essas medidas é igual a 1 (r = 1,0; perfeita), em contrapartida a discordância é evidente, uma vez que o segundo aparelho sempre retorna valores mais elevados.

```
cor(c(80,95,96,100),
    c(90,105,106,110))
[1] 1
```

A ideia de Bland e Altman quando publicaram em 1983 foi avaliar a concordância entre duas variáveis (X e Y) em um gráfico de dispersão, a partir da diferença entre as variáveis (X – Y) e a média das duas (X + Y)/2, par a par de dados. Entenda o princípio por detrás dessa análise: se as duas variáveis concordam, o ideal é que a diferença entre cada par de valor seja zero ou muito próximo de zero. Perceba no exemplo acima que quando aferimos a pressão do primeiro indivíduo deu 80 mmHg por um método, o ideal seria que a aferição com o segundo método desse também 80 mmHg e a diferença entre as duas aferições seria por tanto zero, ou muito próximo disso. No exemplo, a diferença para o primeiro indivíduo foi de 10 mmHg.

Normalmente temos um limite de concordância que vai variar numa faixa de 95% de confiabilidade. Isso implica que no gráfico os limites serão mais ou menos 1,96 vezes o desvio padrão das diferenças. Esse valor remete à distribuição Z.

Vamos usar o pacote blandr para calcular de forma direta as diferenças e as médias par a par, além da construção do gráfico de Bland-Altman. Contudo, há dois passos a mais que iremos realizar para garantir que os levantamentos de dados concordam. Primeiro, vamos realizar um teste t de uma amostra para verificar se a média das diferenças é estatisticamente igual a zero. Quando buscarmos concordância, é o que desejamos que ocorra. Depois, vamos realizar uma regressão linear simples utilizando os resultados das diferenças como variável dependente e as médias par a par como variável independente. Nesse ponto queremos saber se os pontos do gráfico estão homogeneamente distribuídos, pois caso não estiverem há um viés de proporção. Se houver um viés sistemático nas diferenças entre as medidas, isso pode indicar que uma das técnicas de medição tende a fornecer valores mais altos ou mais baixos do que a outra técnica.

Vamos continuar com a ideia de avaliar a concordância entre a aferição da pressão sanguínea sistólica. Uma técnica de enfermagem experiente aferiu a pressão de cinquenta pacientes. Chamaremos de PA1 a aferição por esfigmomanômetro aneroide e de PA2 a aferição por esfigmomanômetro digital. Vamos instalar e abrir a biblioteca blandr e abrir nosso banco de dados chamado "PA_esfig".

```
install.packages('blandr')
library(blandr)

set.seed(54321)
dados <- data.frame(
  PA1 = round(rnorm(50, mean = 120, 12),1),
  PA2 = round(rnorm(50, mean = 120, 10),1)
)

#ou importe o banco de dados
library(rio)
PA_esfig <- import('pasta_de_arquivos/PA_esfig.txt')

head(PA_esfig)
    PA1   PA2
1 117.9 125.0
2 108.9 137.7
3 110.6 118.6
4 100.2 114.7
5 115.1 113.7
6 106.9 132.7
```

Usando a função head() verificamos parte do banco de dados e podemos ver que as aferições sistólicas não são exatamente iguais. Vamos calcular as estatísticas necessárias usando a função blandr.statistics() do pacote blandr. Parte da avaliação é gráfica e para isso vamos utilizar a função blandr.plot.ggplot(); como a própria função demonstra, ela gerará um gráfico utilizando a biblioteca ggplot2 (mais à frente você aprenderá a construir e configurar muitos outros gráficos com ela).

```
#calculando estatística de Bland-Altman
result <- blandr.statistics(PA_esfig$PA1, PA_esfig$PA2)

#Figura 14.1
blandr.plot.ggplot(result)
```

O gráfico gerado será como a Figura 14.1.

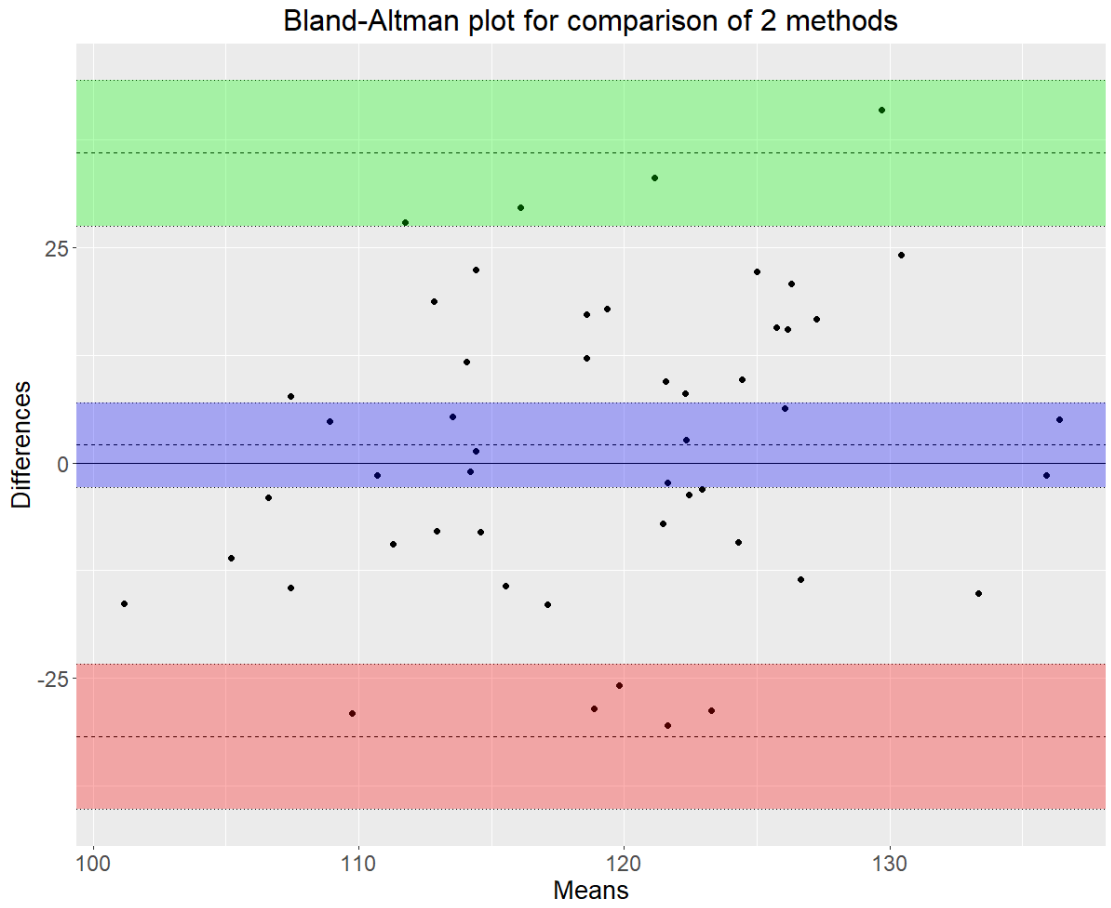


Figura 14.1. Gráfico de Bland-Altman para análise de concordância entre variáveis numéricas.

Primeiramente, a média das diferenças está indicada pela linha horizontal tracejada central. Perceba que ela está bem próxima à linha cheia que marca o valor zero. É o que mais desejamos na questão do nosso exemplo, pois é importante que a média da diferença seja zero ou próximo disso para que haja concordância entre as aferições. As outras duas linhas tracejadas são os limites superior e inferior. Com 95% de confiabilidade que estipulamos, não é desejável que haja pontos que passem desse limite. Quanto mais pontos houverem fora desse intervalo, é mais provável que as duas aferições não estejam concordando entre si. Isso dá a ideia de variabilidade, e se houver uma grande variação nas diferenças entre as medidas, isso pode indicar que uma ou ambas as técnicas de medição têm uma variabilidade maior do que o desejado.

Olhando o gráfico, podemos ver que a grande maioria das diferenças entre as medidas está dentro dos limites de concordância e não há um padrão claro nas diferenças que sugira um viés sistemático ou uma variabilidade maior em uma das técnicas. Isso sugere que as duas técnicas de medição são comparáveis e podem ser usadas de forma intercambiável.

Quando não há concordância entre as duas variáveis, podemos ver padrões como os presentes na Figura 14.2. Perceba que no gráfico A e no gráfico B há a presença de tendências (descendente e ascendente, respectivamente). Outro fato relevante é que os pontos estão mais abaixo do valor zero, mostrando que os dados da segunda variável são na maioria das vezes maiores do que a da primeira variável, pois geram valores negativos majoritariamente. O ideal na concordância de valores é que haja homogeneidade na distribuição dos pontos das diferenças.

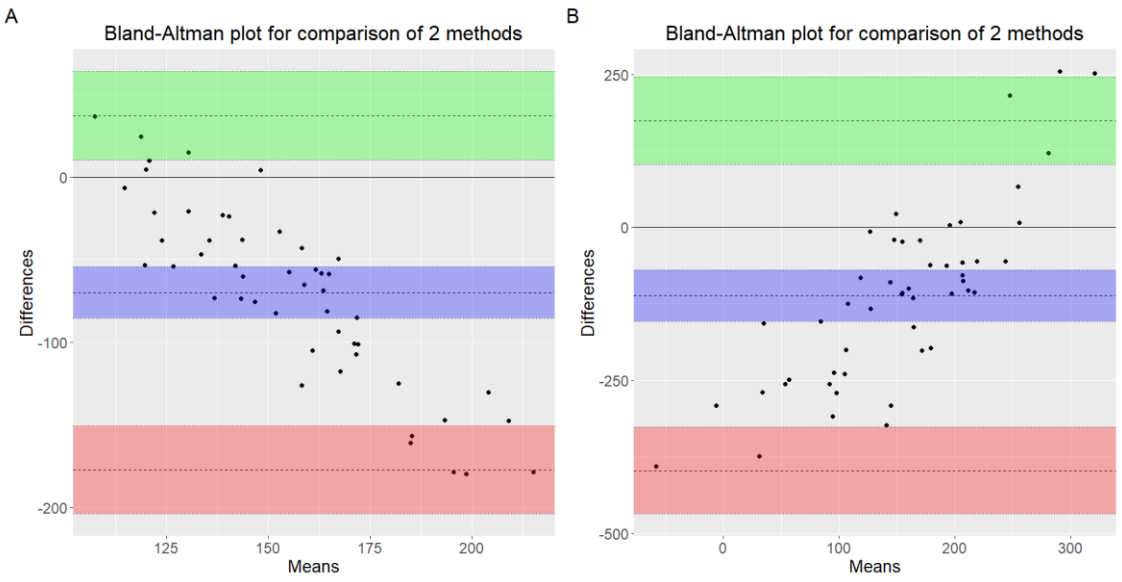


Figura 14.2. Padrões gráficos mostrando falta de concordância entre medições.

Para ter maior confiabilidade em nossa conclusão, vamos realizar o teste t de uma amostra e a regressão linear simples para verificar a homogeneidade das diferenças. Para isso, vamos ter que utilizar parte dos resultados gerados pela função blandr.statistics() que salvamos como "result", apesar de que não precisaríamos ficar presos a esses resultados, pois apenas precisamos dos valores das diferenças entre cada par de dados e a média das diferenças. Vamos usar "result" apenas por praticidade.

```
#adicionando resultados ao banco
PA_esfig$dif <- result$differences     #resultado das diferenças
PA_esfig$medias <- result$means     #resultado das médias
```

Como visto em capítulos anteriores, utilizamos a função t.test() para realizarmos o teste t de *Studant* de uma amostra, para verificarmos a hipótese de que a média das diferenças é igual a zero. Usaremos o parâmetro "mu = 0" para designar ao R a média que se espera igualdade.

A hipótese desse teste t será:

H_0: a média das diferenças é igual a zero.

H_1: a média das diferenças é diferente de zero.

```
t.test(PA_esfig$dif, mu = 0)

      One Sample t-test

data:  PA_esfig$dif
t = 0.85762, df = 49, p-value = 0.3953
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -2.815361  7.007361
sample estimates:
mean of x
    2.096
```

O resultado que mais nos interessa aqui é o seu p-valor = 0,3953. Como ele não foi significativo, aceitamos a hipótese nula de que a média das diferenças é igual a zero. Um grande achado para concluirmos que os dois métodos sejam concordantes.

Caso esse p-valor fosse menor ou igual a 0,05, por tanto significativo, a média das diferenças seriam diferentes de zero, o que nos levaria a concluir que os métodos de aferição não são concordantes.

Para finalizarmos nossos achados, vamos testar a homogeneidade das diferenças em relação à média das diferenças usando uma regressão simples. Para isso vamos usar a função `lm()` aninhada à `summary()` que resume os resultados da regressão.

```
summary(lm(PA_esfig$dif ~ PA_esfig$medias))

Call:
lm(formula = PA_esfig$dif ~ PA_esfig$medias)

Residuals:
    Min      1Q  Median      3Q     Max
-33.793 -10.241   0.049  10.956  33.965

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -53.2502    36.5862  -1.455    0.152
dados$medias   0.4648     0.3066   1.516    0.136

Residual standard error: 17.06 on 48 degrees of freedom
Multiple R-squared:  0.0457,    Adjusted R-squared:  0.02581
F-statistic: 2.298 on 1 and 48 DF,  p-value: 0.1361
```

O resultado que mais nos interessa aqui é o p-valor referente ao beta das médias (P = 0,136). Como ele não foi significativo, podemos dizer que os dados estão homogêneos no gráfico, não havendo indícios de viés de tendência. Mais um achado que comunga na afirmação final de que há concordância entre as aferições da pressão sanguínea pelo esfigmomanômetro aneroide e pelo esfigmomanômetro digital testadas pela enfermeira.

Capítulo 5

Representatividade e Cálculo Amostral

Na bioestatística, as perguntas biológicas são respondidas através dos diversos testes calculados matematicamente, como a gente pode perceber. Para que os resultados sejam confiáveis, precisamos basicamente de qualidade e uma quantidade mínima adequada de dados. O máximo de observações que podemos ter é através do censo, que é a situação onde teríamos os dados de interesse de todos os indivíduos da população.

Contudo, demanda muito tempo e dinheiro para coletar tantos dados em populações reais. Estimativas apontam que o Brasil tenha 203 milhões de pessoas, segundo o Censo 2022. Vamos supor que um pesquisador deseje avaliar a IgE total dos brasileiros. Imaginemos que esse teste laboratorial custe R$ 20,00 cada. Ele teria que desembolsar mais de 4 bilhões de reais para responder a sua pergunta biológica se fosse via censo. Me responda você: é viável?

Prevendo seu bom senso, precisamos de métodos que nos indique uma quantidade mínima de observações onde nós consigamos responder à pergunta biológica, sem perder poder de teste, sem perder a confiança de que provavelmente estejamos certos.

Para isso existem diversas fórmulas para calcularmos o número de amostras adequadas, que chamaremos de **n amostral** a partir de agora, minimamente aceitável para cada tipo de estudo. Mas para isso, precisamos saber qual a pergunta fundamental da pesquisa. Queremos comparar médias? É uma média, duas ou mais de duas? Desejamos relacionar variáveis entre si? Avaliar prevalência na população?

Entenda que calcular o n amostral é parte da fase inicial da pesquisa, mas neste livro, estamos falando sobre mais ao final porque é importante que você tenha tido contato com os vários testes estatísticos para saber primeiramente como você deve responder a sua pergunta biológica. É parte, portanto, do planejamento do projeto calcular o n amostral.

Para seguirmos adiante, precisamos abordar alguns prerrequisitos gerais. Nem todos os testes utilizam um ou outro prerrequisito, mas é bom entender algumas necessidades prévias, como tais:

1. Precisamos saber se a população é finita ou infinita;
2. Precisamos estabelecer os graus de confiança;
3. Precisamos estabelecer o poder do teste;
4. Precisamos de uma média e desvio padrão como base? Fazer estudo piloto.

Vamos aos poucos entender esses pontos. Já discutimos em capítulos iniciai as definições do que é uma população e do que é uma amostra. A Figura 15.1 retoma o assunto demonstrando a diferença de magnitude entre uma e outra. Contudo, agora, precisamos entender as diferentes dimensões entre populações. Por exemplo, a população de trabalho será considerada as pessoas do mundo todo ou de um país específico? Ou de uma cidade? De um logradouro? São tamanhos populacionais bem diferentes. Dependendo de como é feita a pergunta biológica, a população de estudo pode ser infinita ou finita.

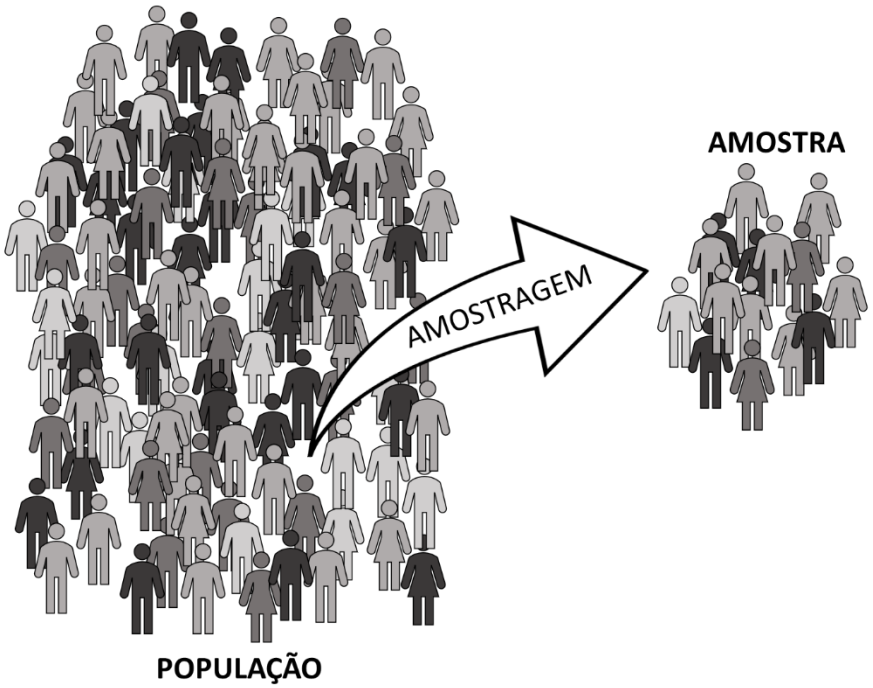


Figura 15.1. Diferença dimensional entre população e amostra

Apesar de que a população brasileira ser realmente finita, em um corte temporal, estamos falando de uma quantidade muito grande de indivíduos. Quando isso ocorre, chamamos estatisticamente de população infinita, porque não factível contabilizá-la rapidamente. Antagonicamente, a população de um determinado prédio residencial é finita, porque é uma quantidade relativamente pequena de indivíduos. Trabalhar com todos os indivíduos residentes em um determinado prédio é factível. Mas, perceba que o conceito entre o que é população infinita e finita é subjetivo. Cabe ao pesquisar se perguntar e decidir qual a dimensão da sua população de estudo.

Mas por que isso é importante? Quando estamos calculando n amostral, por vezes precisamos aplicar uma correção quando a população é considerada finita, pois escolher uma amostra do que já é relativamente pequeno, precisa-se de um fator de correção.

Outro ponto relevante é estabelecermos os graus de confiança e poder do teste. Voltamos ao concento de hipótese estatística e na Tabela 15.1 podemos rever a tomada de decisão segundo os resultados: temos a chance de acertarmos como a de errarmos. Você já sabe que normalmente escolhemos como nível de confiança 95%, dessa forma, prevemos aceitar H_0 com 95% de certeza de que essa hipótese é realmente verdadeira. Contudo, é preciso saber que os testes de hipótese não se preocupam diretamente com a decisão de rejeitar H_0 quando ela realmente pode estar falsa. Esse tipo de preocupação faz parte da amostragem, quando consideramos o poder do teste. Corriqueiramente os pesquisadores tomam um poder de teste de 80%. Uma porcentagem de decisão empírica que equilibra os erros do tipo I e do tipo II.

Tabela 15.1. Tomada de decisão de um teste de hipótese.

	A verdade	
	H_0 é verdadeira	**H_0 é falsa**
Aceitar H_0	**Decisão correta**	**Erro tipo II**
	Nível de confiança (1-α)	(β)
Rejeitar H_0	**Erro tipo I**	**Decisão correta**
	Nível de significância (α)	Poder do teste (1-β)

De forma resumida, quando escolhemos um nível de confiança maior, se torna mais difícil rejeitarmos H_0; o teste fica mais arrojado, conservador. Quando escolhemos um poder de teste maior, maior será o n amostral. Por isso muitos pesquisadores têm na cabeça que quanto maior a quantidade de indivíduos na pesquisa, melhor. Mas nem sempre isso é necessário.

Por fim, o último ponto que devemos levar em conta é a chance da necessidade de realizar um estudo piloto. Especialmente quando a pergunta biológica envolve médias, é preciso ter uma estimativa da média populacional ou mesmo do desvio padrão. Quando temos na literatura algum valor médio e do desvio padrão, podemos utilizar sem problemas essas informações, desde que a população do meu estudo seja semelhante à população que foi avaliada na literatura.

Quando nada tenho em mãos, faz-se necessário realizar um teste piloto, com aproximadamente 30 a 40 indivíduos, para que o cálculo amostral seja adequadamente feito posteriormente. Isso não é um prejuízo para o pesquisador, uma vez que esses indivíduos do estudo piloto podem entrar na pesquisa original, desde que a sua capitação e abordagem tenham sido feitas de forma aleatória. Na realidade, fazer um estudo piloto quando nada se sabe é um investimento. Imagina não realizar um cálculo amostral adequado (ou mesmo não calcular) e no final de toda a pesquisa ela não ser validada por falta de adequação aos métodos científicos.

Podemos ter a melhor pergunta científica do mundo. Se ela não for respondida sistematicamente pelo rigor científico, ela pode ser desprezada e termos problemas na hora da publicação de artigos científicos. Isso sim é um grande prejuízo para quem fez e para a humanidade que poderia se beneficiar desse conhecimento.

Existe uma dúvida comum em instituições de ensino que trabalham com projetos de pesquisa muito extensos, no Brasil comumente em Universidades com diversas colaborações externas. Chamamos de macroprojetos, ou projetos "guarda-chuva", aquelas pesquisas bem amplas, que têm como finalidade contemplar diversos subprojetos. Cada subprojeto gerará conhecimento para um trabalho de conclusão de curso na graduação (TCC), uma dissertação ou uma tese. Geralmente, o macroprojeto tem uma pergunta biológica mais genérica, enquanto que os subprojetos têm perguntas mais específicas.

Sabendo disso, estatisticamente é mais adequado calcularmos os n amostrais de cada subprojeto e definir o n amostral daquele que retornar o maior n. Assim, os demais subprojetos serão contemplados adequadamente.

Para macroprojetos, escolha sempre o maior n amostral, para que a amostragem contemple todos os outros subprojetos.

Muitos dos cálculos amostrais que mostraremos aqui avança um pouco da proposta deste livro, que é focado mais na prática e no entendimento dos resultados. Caso você sinta necessidade de ir além, o que é muito interessante, se aprofunde utilizando a documentação dos pacotes específicos e também com o livro de Jacob Cohen: *Statistical Power Analysis for the Behavioral Sciences*.

Abaixo, trabalharemos o cálculo amostral para cada tido de pesquisa a ser realizada. Para você tomar a decisão de qual cálculo usar, reflita sobre a pergunta biológica e extraia qual será o teste bioestatístico mais importante para responde-la. A partir disso, ande sobre os subcapítulos à seguir.

15.1 QUI-QUADRADO

Se o seu principal objetivo precisa ser respondido por um teste de qui-quadrado, podemos utilizar o pacote pwr para calcular o poder do teste e o n amostral. O poder de teste é uma medida

da capacidade do teste estatístico em detectar um efeito ou diferença estatisticamente significativa, caso ela exista na população de estudo. É a probabilidade de rejeitar corretamente a hipótese nula quando a hipótese alternativa é verdadeira. Em outras palavras, o poder de teste está relacionado à sensibilidade do teste estatístico em detectar um efeito real.

O n amostral está relacionado fortemente com o poder do teste, pois se quisermos uma estatística "poderosa", devemos ter um n amostral adequado. Para isso, alguns parâmetros são necessários para calcularmos o n adequado para os testes que envolvem qui-quadrado: o tamanho do efeito do teste (w), os graus de liberdade (df), o nível de significância (sig.level) e o poder do teste (power).

Usualmente usamos o nível de significância de 5%, o que recai sobre a função o valor de 0,05. O poder do teste é usualmente 80% como já falamos, o que recai na função como valor 0,8. Daqui em diante, sempre que alguma fórmula necessite desses dois parâmetros, utilizaremos esses dois valores para respectivos parâmetros. É um certo consenso na estatística, mas nada disso é rigorosamente obrigatório. Então, vc pode mudá-los a seu gosto. Quanto menor o nível de significância, mais arrojado serão os seus testes. Quanto maior for seu poder de teste, mais arrojado também o será.

O tamanho do efeito "w" em um teste qui-quadrado refere-se à medida de associação entre variáveis categóricas. É uma medida padronizada que representa a força dessa associação. Esse parâmetro pode ser previamente calculado, utilizando-se de dados de um estudo piloto, por exemplo. Contudo, Cohen afirma que podemos utilizar os seguintes valores, dependendo do nível de associação que as variáveis em jogo têm.

- Tamanho do efeito pequeno: w = 0,1.
- Tamanho do efeito intermediário: w = 0,3.
- Tamanho do efeito grande: w = 0,5 ou maior.

Ou seja, quanto mais forte for a associação entre as variáveis, maior será o seu efeito. Perceba que é inversamente proporcional ao p-valor do qui-quadrado que será gerado para a gente chegar nessa conclusão. Quanto menor for o p-valor, mais forte é a associação.

Esse jogo de inversamente proporcional também é verdadeiro para o n amostral que a gente está buscando. Quanto maior for o tamanho do efeito (w), menor será o n amostral.

Por fim, os graus de liberdade será o número de classe que tem juntas as duas variáveis em estudo, subtraído por um (graus de liberdade = número de classes da Var1 + número de classes da

Var2 -1). Portanto, em uma tabela 2x2, teríamos que 2 + 2 – 1 = 3 (três graus de liberdade), por exemplo.

Vamos juntar todos esses parâmetros na função chamada pwr.chisq.test() do pacote pwr. Em nosso exemplo prático, desejamos calcular o número de indivíduos necessários para realizarmos um estudo de associação, via qui-quadrado, entre doença coronariana e o fumo, sendo que ambas estão dicotomizadas (2x2). Por um estudo piloto, observamos que na população de estudo há uma associação intermediária (w = 0,3) e usaremos o nível de significância de 5% e o poder do teste de 80%.

Se você ainda não instalou o pwr, instale-o:

```
install.packages('pwr')
library(pwr)

pwr.chisq.test(w = 0.3, df = 3, sig.level = 0.05, power = 0.8)

     Chi squared power calculation

              w = 0.3
              N = 121.1396
             df = 3
      sig.level = 0.05
          power = 0.8

NOTE: N is the number of observations
```

Perceba que o R retornará o valor de N. Esse é justamente o n amostral que necessitamos, 121,1396. Contudo, não podemos selecionar 121,1396 pacientes. Não existe 0,1396 paciente. Por isso, sempre arredondamos para o valor inteiro maior mais próximo: 122. Ou seja, devemos captar minimamente 122 indivíduos, entre eles com problemas coronarianos e outros não, que haja fumantes e não fumantes.

É comum perdermos informações ao longo da nossa jornada cientifica. Por vezes, o paciente evolui a óbito e não dá para averiguar todos os exames, ou ele deseja simplesmente sair da pesquisa, ou deu algum problema de tabulação de dados e não temos como recuperar os dados de algum paciente, enfim. Podem haver perdas de dados. Para que o n amostral não decaia e você perca o poder de teste que tanto almejou, podemos adicionar 10% (ou mais) ao n amostral calculado.

Seguindo o nosso exemplo, poderíamos fazer: 121,1396 x 1,1 = 133,2536. Arredondando para o número interior maior mais próximo, ficaríamos com 134 pacientes totais.

Ao final de qualquer cálculo amostral, você pode adicionar 10% a 20%, para a prevenção de perdas de observações.

Outra dica: se você já tem em mãos o n amostral e quer saber o poder do seu teste, você deve omitir o parâmetro "power" e adicionar o parâmetro "N" com a quantidade de observações que você tem. Isso é muito comum quando entramos em um projeto de pesquisa abrangente que já está em curso. Nessa situação, normalmente já temos todos os indivíduos da pesquisa captados e iremos responder a uma pergunta que não foi levantada anteriormente no início do projeto maior.

Vamos supor que w continue sendo 0,3, os graus de liberdade de 3 e nível de significância ainda de 5%. Tenho tabulados os dados de 500 indivíduos. Qual o poder do meu teste de qui-quadrado?

```
pwr.chisq.test(w = 0.3, df = 3, sig.level = 0.05, N = 500)

     Chi squared power calculation

              w = 0.3
              N = 500
             df = 3
      sig.level = 0.05
          power = 0.9999825

NOTE: N is the number of observations
```

O poder de teste retornado foi de 99,99%. Extremamente elevado, o que é bom.

Esse tipo de cálculo reverso pode ser aplicado à maioria das funções que iremos utilizar do pacote pwr.

15.2 CORRELAÇÃO

Para estudos em que a análise de correlação seja fundamental para responder à pergunta biológica principal, é necessário calcular o n amostral por outra função. Ainda usando o pacote pwr, utilizaremos a `pwr.r.test()` para esse fim.

Essa função necessita minimamente de três parâmetros: o valor de correlação esperado, o nível de significância, e o poder do teste. Esses dois últimos já padronizamos aqui como: 0,05 e 0,8, respectivamente. Mas e esse coeficiente de correlação?

Bom, esse deve ser feito adequadamente por um teste piloto. Captando poucas amostras aleatoriamente, calculamos o coeficiente de correlação de Pearson e utilizamos na função pwr.r.test().

Se isso não for possível, o plano B é ir até a literatura e buscar um coeficiente de correlação que já foi achado para as duas variáveis que você pretende trabalhar. É um caminho mais fácil, porém, não é dos melhores, pois a população em análise dificilmente será a mesma. Então, fatores intrínsecos à população podem gerar coeficientes de extensão diferentes. Veremos o efeito das dimensões de *r* no n amostral na prática.

Vamos supor que estamos pesquisando a correlação entre a quantidade de eosinófilos (um tipo de célula de defesa presente no sangue) e os níveis de IgE total (anticorpo corriqueiramente associado a alergias ou a infecções helmínticas). Por um teste piloto, achamos a correlação de 0,71 (correlação forte). Vamos calcular o n amostral para essa situação:

```
library(pwr)
pwr.r.test(r = 0.71, sig.level = 0.05, power = 0.8)

     approximate correlation power calculation (arctangh transformation)

              n = 12.36629
              r = 0.71
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

Como resultado, temos n = 12,37, ou seja, bastariam 13 indivíduos para compor a nossa amostra. Quando a correlação é menor, o n amostral tende a aumentar. Vamos testar para *r* = 0,5 e *r* = 0,3:

```
pwr.r.test(r = 0.5, sig.level = 0.05, power = 0.8)

     approximate correlation power calculation (arctangh transformation)

              n = 28.24841
              r = 0.5
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

```
pwr.r.test(r = 0.3, sig.level = 0.05, power = 0.8)

     approximate correlation power calculation (arctangh transformation)

              n = 84.07364
              r = 0.3
```

```
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

Para *r* = 0,5, o n amostral adequado é de 29 indivíduos, enquanto que para *r* = 0,3 é de 85 indivíduos. Isso demonstra que o coeficiente de correlação e o n amostral são inversamente proporcionais nos cálculos desse tipo de amostragem.

Outro ponto a ser lembrado aqui é que correlações podem ser negativas, independentes se fortes ou fracas. O sinal não interfere no cálculo amostral. Vamos exemplificar o cálculo amostral para um coeficiente de correlação igual a -0,64.

```
pwr.r.test(r = -0.64, sig.level = 0.05, power = 0.8)

     approximate correlation power calculation (arctangh transformation)

               n = 15.98746
               r = 0.64
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

O n amostral para uma correlação esperada de -0,64 é de 16 observações.

Se você perceber, se a correlação for muito forte, pouquíssimos indivíduos serão necessários, o que pode gerar uma dúvida: será que essa correlação do estudo piloto realmente satisfaz a minha necessidade ou ela foi gerada ao acaso? Bom, sendo mais conservadores, Cohen sugere o uso do valor de *r* como um tamanho do efeito. Ele lista:

- Efeito baixo: *r* = 0,1;
- Efeito moderado: *r* = 0,3;
- Efeito alto: *r* = 0,5.

Portanto, se a correlação for forte, conserve o valor 0.5 no parâmetro "r" da função pwr.r.test().

15.3 REGRESSÃO LINEAR SIMPLES E MÚLTIPLA

Para se calcular o n amostral cuja pesquisa fundamenta-se em uma regressão linear, seja simples ou múltipla, podemos recorrer ainda ao pacote pwr. Usaremos a função pwr.f2.test() para esse fim.

Nesta função há diversos parâmetros, tais como:

- u = graus de liberdade do numerador;
- v = graus de liberdade do denominador (e de onde tiraremos o n amostral);
- f2 = o tamanho do efeito;
- sig.level = nível de significância;
- power = poder do teste.

O que difere nos cálculos entre as regressões lineares simples e múltiplas fica a cargo dos parâmetros "u" e "v". Os graus de liberdade do numerador (u) é achado usando a quantidade de variáveis totais menos 1 (ou somente a quantidade de variáveis preditoras, se você perceber). Então:

u = número de variáveis independentes

Já os graus de liberdade do denominador (v) é dado pela subtração de N (n amostral, que desejamos saber) pela quantidade de variáveis totais. Então:

v = N – número de variáveis totais

Por aritmética, podemos reescrever a fórmula da seguinte maneira:

N = v + números de variáveis totais

Dessa forma, quando acharmos o valor de v, basta somarmos ao número de variáveis em jogo que encontraremos o nosso n amostral ideal.

Já o parâmetro "f2" pode ser meramente o coeficiente de correlação, ou melhor, a raiz quadrada do coeficiente de determinação que é gerado pela regressão linear, o nosso velho R^2. Entretanto, ainda é melhor assumirmos o R^2 ajustado que é gerado como resultado pelo R. Então:

$$f2 = \sqrt{R^2 ajustado}$$

Cohen propõe níveis diferentes sobre o efeito para essas análises de cálculo amostral para regressões:

- Efeito baixo: f2 = 0,02;
- Efeito moderado: f2 = 0,15;
- Efeito alto: f2 = 0,35.

Se for impossível realizar um teste piloto, mas por conhecimento prévio literário se tenha uma noção do tamanho do efeito das relações entre as variáveis de estudo, utilize esses valores para compor a função no R. Essa parte caba sendo mais subjetiva ao olhar do pesquisador.

Exemplificaremos primeiro o cálculo amostral para uma regressão linear simples. Desejamos prever o comprimento da pétala da flor de íris segundo a sua largura; não importa aqui a espécie. Usaremos o bando iris para isso. Já firmamos que sig.level = 0,05 e power como 0,8. Precisamos achar u, v e R^2 ajustado. Vamos supor, portanto, que esse banco iris não seja o banco final, seja apenas uma representação de um estudo piloto.

```
#exemplo de estudo piloto
data(iris)
regsimples <- lm(Petal.Length ~ Petal.Width, data = iris)

summary(regsimples)

Call:
lm(formula = Petal.Length ~ Petal.Width, data = iris)

Residuals:
     Min       1Q   Median       3Q      Max
-1.33542 -0.30347 -0.02955  0.25776  1.39453

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.08356    0.07297   14.85   <2e-16 ***
Petal.Width  2.22994    0.05140   43.39   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4782 on 148 degrees of freedom
Multiple R-squared:  0.9271,    Adjusted R-squared:  0.9266
F-statistic:  1882 on 1 and 148 DF,  p-value: < 2.2e-16
```

Percebemos aqui que R^2 ajustado foi igual a 0,9266. Vamos calcular o f2 usando a função que calcula a raiz quadrada, sqrt(). Como nós só temos uma variável preditora, então u = 1. Sendo assim, vamos calcular o valor de v:

```
pwr.f2.test(u = 1, f2 = sqrt(0.9266), sig.level = 0.05, power = 0.8)

     Multiple regression power calculation

              u = 1
              v = 8.472015
             f2 = 0.9626006
      sig.level = 0.05
          power = 0.8
```

Então, N = 8,47 + 2 (número de variáveis totais em jogo). Isso resume em 11 observações totais. Apenas onze.

Por que será que esse valor foi relativamente baixo? Devida a relação que o comprimento e a largura das pétalas têm entre si. Vamos testar fazer o mesmo com as sépalas?

```
regsimples2 <- lm(Sepal.Length ~ Sepal.Width, data = iris)
summary(regsimples2)

Call:
lm(formula = Sepal.Length ~ Sepal.Width, data = iris)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5561 -0.6333 -0.1120  0.5579  2.2226

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.5262     0.4789   13.63   <2e-16 ***
Sepal.Width  -0.2234     0.1551   -1.44    0.152
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.8251 on 148 degrees of freedom
Multiple R-squared:  0.01382,   Adjusted R-squared:  0.007159
F-statistic: 2.074 on 1 and 148 DF,  p-value: 0.1519
```

Perceba que não é uma relação significativa estre o comprimento e a largura das sépalas. O R^2 ajustado aqui foi de 0,007159. Portanto:

```
pwr.f2.test(u = 1, f2 = sqrt(0.007159), sig.level = 0.05, power = 0.8)

     Multiple regression power calculation

              u = 1
              v = 92.72376
             f2 = 0.08461087
      sig.level = 0.05
          power = 0.8
```

Sendo assim, nessa situação, precisaríamos de um n amostral de 95 observações.

Em suma, quando a relação é forte, o n amostral não necessita ser grande, ao passo que quando a relação é fraca ou inexistente *a priori*, o n amostral precisa ser grande para evidenciar tal efeito.

Para exemplificar o cálculo amostral para uma regressão linear múltipla, vamos usar as quatro variáveis numéricas do mesmo banco iris. Vamos usar como variável dependente o comprimento da sépala (Sepal.Length). Vamos criar o modelo para vermos quanto ficou o R^2 ajustado.

```
regsimples3 <- lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width, data =
iris)

summary(regsimples3)

Call:
lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width,
```

```
    data = iris)

Residuals:
     Min       1Q   Median       3Q      Max
-0.82816 -0.21989  0.01875  0.19709  0.84570

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.85600    0.25078   7.401 9.85e-12 ***
Sepal.Width   0.65084    0.06665   9.765  < 2e-16 ***
Petal.Length  0.70913    0.05672  12.502  < 2e-16 ***
Petal.Width  -0.55648    0.12755  -4.363 2.41e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3145 on 146 degrees of freedom
Multiple R-squared:  0.8586,    Adjusted R-squared:  0.8557
F-statistic: 295.5 on 3 and 146 DF,  p-value: < 2.2e-16
```

Temos R^2 = a 0,8557. O u é igual a 3 (Sepal.Width, Petal.Length e Petal.Width). Vamos então calcular o valor de v.

```
library(pwr)
pwr.f2.test(u = 3, f2 = sqrt(0.8557), sig.level = 0.05, power = 0.8)

     Multiple regression power calculation

              u = 3
              v = 12.2745
             f2 = 0.9250405
      sig.level = 0.05
          power = 0.8
```

Se N é a soma de v mais o número de variáveis totais, o n amostral sugerido final é de 17 observações, quando arredondamos para cima.

Contudo, pode ser considerado um n amostral pequeno para alguns especialistas. Vamos verificar, portanto, quanto ficaria o n amostral quando usamos o valor de 0,35 para f2, como sugerido por Cohen para tamanho do efeito alto.

```
pwr.f2.test(u = 3, f2 = 0.35, sig.level = 0.05, power = 0.8)

     Multiple regression power calculation

              u = 3
              v = 31.3129
             f2 = 0.35
      sig.level = 0.05
          power = 0.8
```

Sendo assim, N = 31,3129 + 4, que resulta em 35,3129. Arredondando para cima, teríamos o n amostral final de 36 observações. Ou seja, deveríamos ter minimamente 36 flores de íris coletadas aleatoriamente para que possamos fazer a análise de regressão.

15.4 REGRESSÃO LOGÍSTICA

Para calcularmos o n amostral para uma regressão logística iremos usar a função wp.logistic() de um outro pacote, chamado WebPower. Essa função trabalha com alguns parâmetros específicos e dois deles necessitam de um estudo epidemiológico prévio, mas que comumente é de fácil obtenção, na maioria das situações.

Vou exemplificar com uma variável binária como desfecho: controles (não doentes; valor 0) e diabéticos (valor 1). Se na minha pesquisa tiver uma segunda variável independente também binária, como dosagem de glicose com limite de 99 mg/dL, a dicotomia ficaria algo do tipo: menor ou igual a 99 mg/dL é adequado e terá valor 0; maior do que 99 mg/dL é elevado e terá valor 1. Nesse contexto, entre os diabéticos, que é o meu grupo de maior preocupação no estudo, haverá duas proporções: a de apresentar glicose ≤ 99 mg/dL e a de apresentar glicose > 99 mg/mL. Essas duas proporções serão parâmetros fundamentais para calcularmos o n amostral para uma análise de regressão logística.

Chamaremos de p0 a proporção (ou probabilidade) de observar 1 (diabético) para a variável Y quando X for 0 (glicose ≤ 99 mg/mL); Prob(Y=1|X=0). Chamaremos de p1 a probabilidade de observar 1 (diabético) para a variável de resultado Y quando o preditor X é igual a 1 (glicose > 99 mg/dL); Prob(Y=1|X=1). Essas duas proporções nós conseguimos achar em estudos prévios na literatura observando as tabelas de frequência ou devemos fazer um estudo piloto.

Vamos verificar a Tabela 15.2 e utilizá-la como exemplo para nossos cálculos de n amostral para regressões logísticas. Primeiro, observemos que na parte superior (a tabela 2x2) é o formato que costumamos demonstrar as frequências absolutas e relativas do nosso trabalho; não iremos trabalhar diretamente com esses valores. Perceba que as frequências relativas têm como universo a soma das linhas. Entretanto, as proporções que necessitamos de p0 e p1 levam em conta o universo total trabalhado. Exemplificando, a proporção de diabéticos que têm glicose > 99 mg/mL será: 53/203 = 0,2611, aproximadamente. Perceba que 203 é a soma de todos os indivíduos. Dessa forma, encontramos p0 = 0,1330 e p1 = 0,2611.

Tabela 15.2. Tabela de frequência entre diabéticos e suas dosagens categorizadas de glicose sérica de jejum.

Tabela 2x2	**Diabético**	
Glicemia	**Sim (%)**	**Não (%)**
> 99 mg/mL	53 (56,38)	82 (43,62)
≤ 99 mg/mL	27 (24,77)	41 (75,23)
Proporções	**Diabético**	
Glicemia	**Sim**	**Não**
> 99 mg/mL	0,2611[a]	0,2020
≤ 99 mg/mL	0,1330[b]	0,4039

[a] = p1; [b] = p0.

Os demais parâmetros da função que usaremos é o "alpha" (o nível de significância – 0,05), o "power" (poder do teste – 0,8) e "Family" (o tipo de distribuição do preditor), que usaremos "Bernoulli" em nosso exemplo, pois a distribuição de Bernoulli é útil nas variáveis binárias. Caso você esteja usando uma variável preditora numérica que esteja na normal, utilize o argumento "normal". Dê uma olhada na documentação para ver demais possibilidades. Vamos preencher os parâmetros da função na prática. Caso você não tenha instalado o pacote WebPower, faça-o previamente.

```
install.packages('WebPower')
library(WebPower)
wp.logistic(p0 = 0.1330, p1 = 0.2611,
            alpha = 0.05, power = 0.8,
            family = 'Bernoulli')

Power for logistic regression

     p0     p1    beta0     beta1        n alpha power
  0.133 0.2611 -1.87469 0.8344307 312.3767  0.05   0.8
```

Segundo os resultados, necessitaríamos de 313 indivíduos no total (n = 312,3767). Dividiríamos 157 com o desfecho e 156 como controles, ou vice-versa.

Um fato interessante é que quanto maior for a diferença entre p0 e p1, menor será o n amostral. Isso é racionalmente compreensível, pois quanto mais os dois grupos se diferem em relação à variável independente, não necessitaríamos de grande amostragem para explicar tal diferença entre as proporções.

Quando temos proporções mais próximas, explicar a diferença entre os grupos se torna uma tarefa mais complicada. Assim, necessitaremos de uma maior quantidade de indivíduos em jogo. Em estudos genéticos isso é muito comum, pois quando estamos comparando frequências de determinados genótipos na mesma população, mas de desfechos diferentes, a genética comumente é bastante similar. O que correrá é que para avaliações genéticas precisaríamos de uma quantidade maior de indivíduos para tentar explicar diferenças de proporções genotípicas.

De forma grosseira, vamos verificar o que ocorre quando p0 e p1 variam entre si.

Exemplo 1: p0 = 10% e p1 = 80%.

```
wp.logistic(p0 = 0.1, p1 = 0.8,
            alpha = 0.05, power = 0.8,
            family = 'Bernoulli')
Power for logistic regression

     p0  p1     beta0    beta1        n alpha power
    0.1 0.8 -2.197225 3.583519 21.22241  0.05   0.8
```

Resultado: n amostral igual a 22 indivíduos totais.

Exemplo 2: p0 = 10% e p1 = 50%.

```
wp.logistic(p0 = 0.1, p1 = 0.5,
            alpha = 0.05, power = 0.8,
            family = 'Bernoulli')
Power for logistic regression

     p0  p1     beta0    beta1        n alpha power
    0.1 0.5 -2.197225 2.197225 49.13422  0.05   0.8
```

Resultado: a amostral igual a 50 indivíduos totais.

Exemplo 3: p0 = 10% e p1 = 20%.

```
wp.logistic(p0 = 0.1, p1 = 0.2,
            alpha = 0.05, power = 0.8,
            family = 'Bernoulli')
Power for logistic regression

     p0  p1     beta0     beta1        n alpha power
    0.1 0.2 -2.197225 0.8109302 414.4262  0.05   0.8
```

Resultado: n amostral de 415 indivíduos totais.

Em suma, quanto maior a diferença entre p0 e p1, menor será o n amostral retornado. Enquanto que, quanto menor a diferença entre p0 e p1, maior será o n amostral.

15.5 TESTE T PARA UMA AMOSTRA

Para avaliar o tamanho mínimo adequado do n amostral de estudo que deseje comparar uma média amostral a uma média já estabelecida, podemos usar o pacote pwr, através da função pwr.t.test().

Os parâmetros que devemos informar ao R são: "d" (tamanho do efeito), "sig.level" (nível de significância), "power" (poder do teste) e "type" (o tipo de teste a ser usado; será "one.sample" – para uma amostra). Para o tamanho do efeito considere:

- valor 0.2 para um pequeno efeito;
- valor 0.5 para um médio efeito;
- valor 0.8 para um grande efeito.

Lembrando que o tamanho do efeito exprime a magnitude sob a hipótese alternativa. Dessa forma, quanto maior o tamanho do efeito, mais fácil é detectar um efeito e isso requer menos amostras. Em outras palavras: o tamanho do efeito é inversamente proporcional ao tamanho amostral e ao poder de teste.

Na comparação de uma média é mais difícil estabelecer um tamanho de efeito sem um estudo piloto prévio. A dica é testar os vários tamanhos de efeitos e definir pelo menor efeito ao qual trará o maior tamanho amostral factível para a sua pesquisa. Equilíbrio é a chave. Por isso, nos exemplos a seguir usaremos um efeito de tamanho médio.

Como exemplo, vamos supor que um clínico deseja avaliar se os seus pacientes têm a mesma média de dosagem de colesterol sérica com a média mundial. Você perceberá que para o cálculo amostral de uma média é o mais simples possível. O cálculo não depende de uma média prévia nem do seu desvio padrão. Por isso, vamos avaliar quantos pacientes esse clínico precisa minimamente para responder a sua pergunta biológica. Para isso, vamos pôr como tamanho do efeito o valor de 0,5, nível de significância de 5%, poder do teste de 80% e o tipo será de uma amostra.

```
library(pwr)
pwr.t.test(d = 0.5,
           sig.level = 0.05,
           power = 0.8,
           type = 'one.sample')
     One-sample t test power calculation

              n = 33.36713
              d = 0.5
      sig.level = 0.05
```

```
          power = 0.8
    alternative = two.sided
```

Como resultado, podemos ver que o n amostral mínimo necessário é de 34 pacientes (n = 33,37). Se ele deseja-se ter um tamanho de efeito baixo (0,2), o que é aconselhável testar, ele deveria selecionar 199 pacientes.

```
pwr.t.test(d = 0.2,
           sig.level = 0.05,
           power = 0.8,
           type = 'one.sample')
     One-sample t test power calculation

              n = 198.1508
              d = 0.2
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

15.6 TESTE T INDEPENDENTE

Para a avaliação do n amostral adequado para a comparação de duas médias independentes é interessante que conhecemos as médias e desvios padrão dos dois grupos de interesse, pois precisaremos estimar o tamanho do efeito. Aqui, ele é calculado pela expressão:

$$D = \frac{\bar{x}2 - \bar{x}1}{S}$$

Na equação, D é o tamanho do efeito, que servirá para preencher um dos parâmetros da função do R que utilizaremos em breve; $\bar{X}2$ e $\bar{X}1$ são as médias de cada grupo extraídas da literatura ou de um estudo piloto; e S é o desvio padrão combinado (no inglês, você pode achar como *pooled standard deviation*). Esse desvio padrão combinado pode ser meramente a média aritmética dos respectivos desvios padrões de cada grupo.

Caso seja impossível estimar o tamanho do efeito, podemos recorrer aos pontos de corte que Cohen estabeleceu:

- Efeito pequeno: 0,2;
- Efeito de tamanho médio: 0,5;
- Efeito grande: 0,8.

Lembrando que quanto menor for o efeito, maior será o n amostral. Portanto, são grandezas inversamente proporcionais.

No R, trabalharemos com a mesma função usada mais acima: pwr.t.test() do pacote pwr. Nessa função devemos fornecer valores para os seguintes parâmetros: "d" (tamanho do efeito), "sig.level" (nível de significância – 0,05), "power" (poder do teste 0,8), e "type" (tipo de teste) que será para duas amostras (two.sample). A função retornará, dentre outras coisas, o valor de n, que é o tamanho amostral para cada grupo.

Em nosso exemplo prático, vamos calcular o tamanho do efeito e avaliaremos por fim o tamanho amostral. Sendo assim, vamos supor que um pesquisador esteja avaliando a diferença da dosagem sérica de ferritina (um marcador bioquímico importante na avaliação de anemias) em dois grupos de pacientes (A e B). Em um estudo piloto, o pesquisador encontrou uma média de 450 µg/L de ferritina no grupo A, com desvio padrão de 53 µg/L. No grupo B, ele encontrou média de 410 µg/L, com desvio padrão de 51 µg/L. Ele deseja estimar o n amostral para uma pesquisa maior cujo principal objetivo está na comparação da ferritina nesses dois grupos. Para isso, ele estabeleceu o nível de significância de 5% e poder de teste de 80%. Como calcular o n amostral no R?

```
library(pwr)
#cálculo do tamanho do efeito
D <- (450-410)/mean(c(53,51))
D
[1] 0.7692308

#cálculo do n amostral
pwr.t.test(d = D,
           sig.level = 0.05,
           power = 0.8,
           type = 'two.sample')

     Two-sample t test power calculation

              n = 27.52331
              d = 0.7692308
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number in *each* group
```

Primeiramente, calculamos o tamanho do efeito, onde o R retornou o valor aproximado de 0,77, que segundo Cohen, é um efeito de tamanho considerado médio. Já o cálculo do tamanho amostral, o R retornou o valor 27.5233, ou seja, necessita-se de minimamente 28 indivíduos **em cada grupo**. Portanto, a pesquisa precisa de 56 participantes no total.

15.7 TESTE T PAREADO

Na avaliação do n amostral para a comparação de médias pareadas se vale da mesma função que trabalhamos anteriormente. Inclusive o cálculo do tamanho do efeito é exatamente o mesmo. Por isso, estudos prévios podem ser valiosos.

Indo direto ao nosso exemplo no R, vamos imaginar que uma determinada enfermeira esteja fazendo um trabalho de conclusão de curso de especialização. Ela está avaliando a gradação da dor epigástrica ao mudar o posicionamento do paciente com problemas gástricos graves no leito. Inicialmente os pacientes encontram-se em decúbito dorsal (deitado com barriga para cima) e é avaliado sua dor em uma escala de 0 a 10. Troca-se a posição para decúbito lateral (ficar deitado de lado) e depois de 30 minutos ela volta e reavalia a escala de dor. A sua hipótese sugere que pacientes com dor epigástrica têm alívio da dor ao ficar em decúbito lateral.

Perceba uma coisa fundamental nessa situação: a enfermeira estará avaliando os mesmos pacientes, mas em momentos diferentes, utilizando-se de uma variável numérica. Por isso, um teste importante para ela responder a sua pergunta biológica certamente será o teste t pareado ou um equivalente não paramétrico.

Ela poderia fazer um estudo piloto para avaliar o tamanho amostra de forma mais adequada, mas ela não tem tempo disponível para tal. Ela preferiu estimar o n amostral através dos pontos de corte estabelecidos por Cohen para o tamanho do efeito:

- Efeito pequeno: 0,2;
- Efeito de tamanho médio: 0,5;
- Efeito grande: 0,8.

Ela estima um efeito de tamanho médio. O que devemos preencher como argumento na função `pwr.t.test()` é o tamanho do efeito, nível de significância de 5% e poder de teste de 80% e o tipo de teste no parâmetro "type" pôr como "paired". Com isso, o R retornará a quantidade de pares de observações, que na situação dela é a quantidade de pacientes a ser analisada.

```
pwr.t.test(d = 0.5,
           sig.level = 0.05,
           power = 0.8,
           type = 'paired')

     Paired t test power calculation

              n = 33.36713
              d = 0.5
      sig.level = 0.05
```

```
          power = 0.8
    alternative = two.sided

NOTE: n is number of *pairs*
```

Como resultado, ela terá um n amostral de 33.37 pares de observações, ou seja, 34 pacientes a estudar.

15.8 ANOVA DE UMA VIA

Quando o nosso trabalho precisa responder fundamentalmente uma pergunta cuja resposta se baseará em uma comparação de três ou mais médias, devemos recorrer a Análise de Variância (ANOVA). Para estimarmos o n amostral adequado nessa situação podemos usar a função pwr.anova.test() do pacote pwr.

Na referida função devemos atribuir valores aos parâmetros: “k” (número de grupos), “f” (tamanho do efeito), “sig.level” (nível de significância) e “power” (poder do teste). Sobre o tamanho do efeito não exemplificarei seus cálculos aqui, pois ultrapassa o escopo da complexidade do livro. Contudo, caso você deseje se aprofundar, sugiro estudar tal método através do livro de Cohen (1988) já citado.

Entretanto, Cohen sugere três limites para utilizarmos no parâmetro “f”:

- Tamanho do efeito pequeno: 0,1;
- Tamanho do efeito médio: 0,25;
- Tamanho do efeito grande: 0,4.

Lembrando que o tamanho do efeito está relacionado à hipótese alternativa do teste. Assim, quanto mais fácil for a detecção de diferenças entre as médias, maior será o tamanho do efeito. Como avaliaríamos isso em um estudo piloto ou utilizando um artigo como base? Ao achar o valor de P para o teste F (da ANOVA), se ele for muito significativo, ou seja, menor do que 0,01, é sugestivo de um efeito de tamanho grande. Se for próximo de 0,05 ou não significativo, podemos sugerir um tamanho de efeito médio a pequeno.

Para exemplificarmos isso no R, vamos supor que um psicólogo esteja avaliando três grupos de indivíduos que trabalham em setores diferentes de empresas de um determinado segmento: vendedores, estagiários do administrativo e profissionais do financeiro. Ele aplicará um questionário de qualidade de vida, cujo resultado final será um valor numérico (um escore). Assim, ele terá a média para cada grupo e o seu intuito é compará-las.

Por experiência prévia, ele avalia que o tamanho do efeito é médio, pois foi estimado que não há tanta diferença entre a qualidade de vida dos vendedores e o pessoal do financeiro. O poder do teste será 80% e o nível de significância será de 5%. Com essas informações, vamos para o R.

```
library(pwr)
pwr.anova.test(k = 3, f = 0.25, sig.level = 0.05, power = 0.8)
    Balanced one-way analysis of variance power calculation

              k = 3
              n = 52.3966
              f = 0.25
      sig.level = 0.05
          power = 0.8

NOTE: n is number in each group
```

Perceba o R retorna o valor de n, que é o nosso n amostral. Ele indica a quantidade de participantes em cada grupo. Portanto, o psicólogo precisará em sua avaliação de 53 participantes em cada grupo, ou seja, 159 indivíduos totais.

15.9 OUTROS MÉTODOS DE ESTIMAÇÃO DO N AMOSTRAL

Se nenhum dos exemplos acima servir para o seu propósito, sugiro que você busque alternativas lendo a documentação dos pacotes pwr e WebPower, primeiramente. Neles, haverá outras funções que retornarão o n amostral para outros tipos de testes estatísticos. Não só isso, a variação dos parâmetros que utilizamos servirá para responder problemas sensivelmente diferentes.

Existem outras formas para calcular o n amostral para as situações apresentadas no capítulo e outras mais. Sugiro a leitura do artigo escrito por Miot (2011). Nele, você encontrará alternativas matemáticas para calcular o n amostral da maioria das situações clínico-epidemiológicas, considerando inclusive populações finitas e infinitas.

Outra alternativa é acessar o site PPGCS Sample Size (www.oraculoestatistico.com.br/ppgcssamplesize/public), que apesar do nome está em português. Nele você encontrará métodos outros para estimar o tamanho amostral, inclusive para comparações de proporções. Ele é mantido e atualizado por professores da Universidade Estadual de Montes Claros.

Por fim, sugiro aprofundamento através de dois livros. O primeiro é o de Cohen (*Statistical power analysis for the behavioral sciences*), que pode ser complexo para iniciantes no tema, contudo, ele é bem abrangente no quesito poder de teste e determinação do tamanho amostral. O segundo livro é o de Bolfarine (Elementos de amostragem). Mais amigável e em português.

Capítulo

16

Teste Diagnóstico

A utilização de testes diagnósticos desempenha um papel crucial na medicina moderna, permitindo a avaliação objetiva da presença ou ausência de condições de saúde em indivíduos. Entre as ferramentas mais comuns nesse contexto, as tabelas 2x2 se destacam pela sua facilidade de interpretação. A avaliação da eficácia dos testes diagnósticos é um campo de grande interesse na investigação clínica e na epidemiologia. O termo "teste diagnóstico" engloba não apenas exames laboratoriais, mas também procedimentos de avaliação clínica, exame físico e uma variedade de métodos inclusive terapêuticos.

Neste contexto, exploraremos como as tabelas 2x2 fornecem uma estrutura para a análise estatística de acurácia, sensibilidade, especificidade, valores preditivos e outros parâmetros cruciais que auxiliam os profissionais da saúde na tomada de decisões.

16.1 A TABELA 2 X 2

Médicos e médicos veterinários estão diariamente diagnosticando queixas a partir de sinais, sintomas e exames laboratoriais ou acessórios. Essa determinação do diagnóstico é sempre um processo imperfeito, até mesmo porque na área biológica 1 + 2 é igual a “depende da situação”. Essa imperfeição vem de diversos fatores como casualidade, vieses diversos, contudo, acredito que o mais importante é que a complexidade dos organismos vivos é tamanha que se torna impossível recorrer a todas as variáveis biológicas possíveis para ter 100% de certeza em uma tomada de decisão. São muitas e complexas reações ocorrendo ao mesmo tempo, que torna impossível dizer que todas elas estão em harmonia e na mesma intensidade em todos os seres vivos de mesma espécie.

O resultado do diagnóstico é na realidade uma probabilidade de estar certo. Entenda que todo método laboratorial, por exemplo, seja de dosagem de um marcador bioquímico com a glicose ou seja um exame qualitativo como o teste do HIV, tem um erro intrínseco. Então, perceba que tanto homem quanto máquina podem em algum grau errar um certo diagnóstico.

O que há de mais importante não é exatamente o erro em si, mas avaliar e diminuir o máximo possível a chance de estar errado. Para isso foi criado critérios avaliativos dentro de uma tabela simples, a tabela 2x2 ou tabela de contingência 2x2. Nela há quatro valores, preenchidos por duas variáveis categóricas (ou que foram categorizadas) e dicotômicas (Tabela 16.1). Colocamos na coluna a variável chamada **padrão-ouro** e na linha a variável chamada **teste**. Vou explicar

o que corresponde cada uma dando alguns exemplos, mas é preciso que você, com a sua maturidade no assunto, expanda esse conhecimento em outras aplicações.

Tabela 16.1. Tabela de contingência 2x2.

		Padrão-ouro	
		Positivo	Negativo
Teste	Positivo	Verdadeiro positivo (VP ou a)	Falso positivo (FP ou b)
	Negativo	Falso negativo (FN ou c)	Verdadeiro negativo (VN ou d)

Pois bem, o termo padrão-ouro se refere a um método, um teste, procedimento ou critério amplamente aceito como a referência mais confiável e precisa para determinar a presença ou ausência de uma condição de interesse, como uma doença. O padrão-ouro é usado como uma comparação para avaliar a precisão e a validade de outros métodos de diagnóstico.

Apesar de mais confiável, na prática, há situações em que não é possível ter um padrão-ouro absoluto, devido a complexidades da doença em si, falta de conhecimento completo sobre sua condição ou limitações tecnológicas.

Por vezes, o padrão-ouro pode ser um conjunto de técnicas ou um exame que foi testado duas vazes: uma duplicata, onde o teste foi avaliado duas vezes no mesmo momento; ou, um reteste, cujo exame foi feito duas vezes em momentos diferentes. Para o diagnóstico do HIV (Vírus da Imunodeficiência Humana), alguns hospitais combinam o teste de ELISA (*Enzyme-Linked Immunosorbent Assay*) com o Western Blot para fechar o diagnóstico da infecção; juntos são um padrão-ouro. Contudo, temos os testes rápidos para HIV-1 e HIV-2, que servem como triagem da população.

Já o teste é um exame ou critério de avaliação secundário ou novo, que por ventura precise ser avaliada a precisão dos resultados vindos dele. Ou seja, precisa ser validado.

No diagnóstico da tuberculose, uma doença infecciosa bacteriana, o padrão-ouro é o exame chamado BAAR (bacilo álcool-ácido resistente), que consiste em verificar a presença da bactéria através da microscopia, fazendo uma coloração específica da lâmina de esfregaço. Contudo, pode-se fazer uso de testes rápidos para a detecção da bactéria. Algumas vezes, esses testes rápidos

podem não ser tão precisos quanto os padrões-ouro respectivos. Por isso, são colocados como teste em seu processo de validação.

Voltando à Tabela 16.1, perceba que ela é composta por quatro células. Existe a possibilidade de o padrão-ouro dar um resultado positivo e o teste também dar um resultado positivo. A essa concordância damos o nome de verdadeiros positivos. A concordância pode haver entre os negativos do padrão-ouro e do teste, o qual chamamos de verdadeiros negativos. Quando o padrão-ouro gera um resultado positivo e o teste gera um resultado negativo, a essa discordância chamamos de falso negativo. Por fim, quando o padrão-ouro gera um resultado negativo e o teste um resultado positivo, temos um falso positivo.

Muitos autores gostam de resumir essas quatro células em letras: a, para os verdadeiros positivos; b, para os falsos positivos; c, para os falsos negativos; d, para os verdadeiros negativos. Isso ajudará muito na construção e entendimento das fórmulas que iremos aplicar aqui. Mas, perceba que essa atribuição dependente exatamente de como a tabela foi montada. Se invertermos a ordem positivo e negativo na tabela, tudo mudará. Alguns autores preferem lidar com as letras iniciais dos nomes de cada célula: VP, FP, FN e VN. O valor total, ou seja, a soma de todos os indivíduos que estão na tabela, podemos chamar de n (nosso n amostral), que na verdade é a soma de a + b + c + d.

A Tabela 16.2 apresenta valores nas quatro células que compõe uma tabela 2x2 fictícia. Podemos observar que há 15 observações que são verdadeiros positivos, 2 são falsos positivos, 1 é falso negativo e 16 são verdadeiros negativos. Verifique que o total de observações foi 34. O padrão-ouro retornou 16 positivos e 18 negativos totais. Já o teste, retornou 17 positivos e 17 negativos.

Tabela 16.2. Tabela de contingência 2x2 com valores.

		Padrão-ouro	
		Positivo	Negativo
Teste	Positivo	15	2
	Negativo	1	16

Fica evidente que o suposto teste não foi perfeito em comparação ao padrão-ouro utilizado. Mas ele pode ser considerado um bom teste? Para responder essa pergunta precisamos avaliar alguns parâmetros que podem ser calculados a partir da nossa tabela 2x2. São eles: acurácia,

sensibilidade, especificidade, valor preditivo positivo, valor preditivo negativo, entre outros. Vamos esmiuçar tudo o que essa tabela de contingência pode nos oferecer.

16.2 ACURÁCIA

A acurácia do teste é a porcentagem, ou probabilidade, de acerto do teste no total. Podemos dizer também que é a porcentagem onde os exames comungam. Na tabela de diagnóstico 2x2, onde o teste acerta? Nos verdadeiros positivos e nos verdadeiros negativos. O total é n, ou a soma de todas as células da tabela. Dessa forma, podemos calcular facilmente a acurácia do teste usando a fórmula:

$$Acurácia = \frac{a + d}{a + b + c + d}$$

Nessa parte dos nossos estudos abordaremos muitas fórmulas, mas eu encorajo você a não as gravar de início. Entenda a essência do parâmetro e calcule sem precisar memorizar suas fórmulas. Na Tabela 16.2 podemos calcular a acurácia identificando as células: a = 15 e d = 16. Então,

$$Acurácia = \frac{15 + 16}{15 + 2 + 1 + 16} = \frac{31}{34} = 0{,}9118$$

Quando multiplicamos esse valor por cem, retornará o valor em porcentagem, 91,18%. Ou seja, o teste acerta 91,18% das vezes. No seu cálculo original, a acurácia varia entre 0 e 1, e quanto mais próximo de 1, mais o teste se equipara ao padrão-ouro.

O antagônico da acurácia é chamado de taxa de erro, que a porcentagem que o teste erra. Esse valor pode ser encontrado diminuindo 1 da acurácia encontrada na forma decimal, ou (b + c) / n. Em nosso exemplo foi 0,0882, ou 8,82%.

16.3 SENSIBILIDADE

A sensibilidade é a proporção de indivíduos doentes, ou uma condição, que apresentaram positivo para o teste. Ou seja, dentre os positivos do padrão-ouro, quantos deram também positivos no teste. A fórmula para calcular a sensibilidade é:

$$Sensibilidade = \frac{a}{a + c}$$

Se você perceber que estamos restringindo em responder a porcentagem de acerto entre os positivos do padrão-ouro, você matou a charada. Calculando, portanto, a sensibilidade através da Tabela 16.2, teremos:

$$Sensibilidade = \frac{15}{15+1} = \frac{15}{16} = 0{,}9375$$

Dessa maneira, podemos dizer que a sensibilidade no nosso exemplo foi de 0,9375 ou 93,75%. Assim como ocorre com a acurácia, na forma decimal o espectro da sensibilidade varia de 0 a 1. Quanto mais próximo de 1 melhor é, mais comunga com o padrão-ouro.

Em outros contextos, a sensibilidade pode ser chamada também de *recall*.

16.4 ESPECIFICIDADE

A especificidade é a proporção de doentes que testaram negativo entre todos os negativos do padrão-ouro, ou seja, a proporção de indivíduos sem a doença cujo teste foi negativo.

Comumente temos a tendência de dar mais valor à sensibilidade, que é a detecção dos positivos. Entretanto, a especificidade é tão importante quanto. Imagine que exista um teste inovador cujas testagens sejam todas positivas. Devemos medicar todo mundo que foi testado? Obvio que não. Sensibilidade e especificidade ando juntas. Dependendo do tipo da doença é que podemos dar um pouco mais de ênfase a uma ou a outra.

Para calcular a especificidade, usamos a fórmula:

$$Especificidade = \frac{d}{b+d}$$

Calculando a especificidade referente a Tabela 16.2:

$$Especificidade = \frac{16}{2+16} = \frac{16}{18} = 0{,}8889$$

Obtivemos uma especificidade de 0,8889, ou seja, entre os saudáveis o teste confirmou negativo em 88,89% deles.

16.5 VALOR PREDITIVO POSITIVO

O valor preditivo positivo (VPP) trata-se da probabilidade de o paciente estar doente sendo que seu teste deu positivo. Ou, podemos dizer ainda que é a proporção dos realmente doentes dentre todos os indivíduos que o teste retornou como doentes (positivos).

A fórmula que costumamos utilizar para calcular facilmente o VPP é dada por:

$$VPP = \frac{a}{a+b}$$

Calculando o valor preditivo positivo da Tabela 16.2, teremos:

$$VPP = \frac{15}{15+2} = \frac{15}{17} = 0{,}8824$$

Ou seja, nosso VPP foi de 0,8824, ou podemos dizer que a probabilidade de acharmos um positivo dentro dos resultados positivos do teste é de 88,24%. Para que fique o mais claro possível, isso quer dizer que: se Fulano obteve um exame cujo o resultado foi positivo, a probabilidade dele estar realmente doente é de 88,24%.

Em outros contextos, o valor preditivo positivo pode ser chamado também de precisão.

16.6 VALOR PREDITIVO NEGATIVO

O valor preditivo negativo (VPN) é a probabilidade de não estar doente, sendo que o teste deu negativo. Ou seja, é a proporção dos saudáveis dentre todos os indivíduos que o teste retornou como negativo.

A formula para o VPN é:

$$VPN = \frac{d}{c+d}$$

Em nosso exemplo, temos que:

$$VPN = \frac{16}{1+16} = \frac{16}{17} = 0{,}9412$$

Assim, podemos dizer que a proporção dos saudáveis entre todos os que o teste retornou como negativos foi de 94,12%. Então, podemos dizer que se Fulano fez um exame e esse deu negativo, a probabilidade de ele não estar doente é de 94,12%.

16.7 RAZÃO DE VEROSSIMILHANÇA POSITIVA

Usamos também as razões de verossimilhança para descrever o desempenho do teste diagnóstico. Elas têm um sentido interpretativo levemente diferente de probabilidade, apesar de estarem intimamente relacionadas. Na realidade, o seu entendimento depende do que conhecemos como chances (*odds*). A chance é fruto da razão entre duas probabilidades: a probabilidade de um evento ocorrer e a probabilidade de o evento não ocorrer (probabilidade complementar). Essa probabilidade complementar é meramente 1 menos a probabilidade de o evento ocorrer. Podemos escrever matematicamente a chance da seguinte maneira:

$$Chance = \frac{Probabilidade\ do\ evento\ ocorrer}{Probabilidade\ do\ evento\ não\ ocorrer}$$

$$Chance = \frac{Probabilidade\ do\ evento}{(1 - Probabilidade\ do\ evento)}$$

Vamos supor que um lutador tem a chance de 6:1 de ganhar o cinturão de luta livre dos pesos pesados (forma a qual normalmente a gente escuta em casas de aposta). A partir dessa informação, podemos saber qual a probabilidade do lutador ganhar segundo a visão dos apostadores.

$$Chance = \frac{Probabilidade}{1 - Probabilidade}$$

$$\frac{6}{1} = \frac{Probabilidade}{1 - Probabilidade}$$

$$6 - 6.Probabilidade = Probabilidade$$

$$6 = 7.Probabilidade$$

$$Probabilidade = \frac{6}{7} = 0{,}8571$$

Ou seja, podemos dizer que a probabilidade do lutador ganhar é de 85,71%, segunda as casas de aposta. Se a chance foi 9:1, a probabilidade de ele ganhar é de 90,00%. Se a chance for

de 15:1, a probabilidade de ganhar é 93,75%. Quanto maior o numerador, mais próximo de 100,00% a probabilidade será.

Especificamente falando, a razão de verossimilhança positiva (*RV+*) trata-se da razão entre a proporção de pessoas doentes que têm um resultado de teste positivo (sensibilidade) e a proporção de pessoas que não estão doentes cujo teste também é positivo (1 - especificidade).

$$RV+ = \frac{\frac{a}{a+c}}{\frac{b}{b+d}} = \frac{Sensibilidade}{1 - Especificidade}$$

Exemplificando, vamos retornar à Tabela 16.2. Calculando a razão de verossimilhança positiva, temos:

$$RV+ = \frac{\frac{15}{15+1}}{\frac{2}{2+16}} = \frac{\frac{15}{16}}{\frac{2}{18}} = 8{,}4375$$

Assim, podemos interpretar esse valor da seguinte maneira: a probabilidade de ter uma pessoa doente é 8,44 vezes maior do que a probabilidade de uma pessoa não ter a doença quando o resultado do teste é positivo. Exemplo prático: um teste de gravidez tem RV+ igual a 15,46; isso indica que quando um teste for positivo a probabilidade da mulher está grávida é 15,46 vezes maior do que a probabilidade de não estar.

Esse "maior" que foi usado na frase na verdade é redundante e sugiro até que seja omitido, pois qualquer valor vezes um número maior do que 1 (exemplo: 1,5) é maior do que o valor original. Mas ele pode caber na oração se o escritor desejar enfatizar o que se deseja passar. É talvez um pleonasmo.

Se você percebeu o sentido interpretativo da razão de verossimilhança positiva, então você também entende que quanto maior a RV+, melhor será o teste a nível discriminativo.

16.8 RAZÃO DE VEROSSIMILHANÇA NEGATIVA

Razão de verossimilhança negativa (RV-) é a razão entre a proporção de pessoas doentes com resultados negativos (1 – sensibilidade) e a proporção de pessoas não doentes com resultado de teste negativo (especificidade).

$$RV- = \frac{\frac{c}{a+c}}{\frac{d}{b+d}} = \frac{1 - Sensibilidade}{Especificidade}$$

Vamos recorrer mais uma vez à Tabela 16.2 para exemplificar seu cálculo:

$$RV- = \frac{\frac{1}{15+1}}{\frac{16}{2+16}} = \frac{\frac{1}{16}}{\frac{16}{18}} = 0{,}0703$$

A RV- é uma medida que nos ajuda a entender como é provável que um resultado negativo no teste de triagem esteja associado à ausência real da doença. Neste caso, uma RV- de 0,0703 indica que a probabilidade de um paciente não estar doente, dado um resultado negativo no teste, é cerca de 0,0703 vezes menor do que a probabilidade de um paciente estar doente.

Em outras palavras, uma RV- menor que 1 sugere que o teste é informativo na exclusão da doença quando o resultado do teste é negativo. Dessa forma, quanto menor for o RV-, melhor será o teste.

16.9 PREVALÊNCIA

Abordaremos melhor a ideia de prevalência de uma doença em capítulos seguintes, adentrando mais na epidemiologia. Contudo, quando estamos analisando dados a partir das tabelas 2x2, podemos dizer basicamente que a prevalência é a proporção de pessoas doentes determinada pelo padrão-ouro no tempo decorrido do estudo.

Retomando a Tabela 16.1, encontraremos a fórmula para a prevalência da seguinte maneira:

$$Prevalência = \frac{a+c}{a+b+c+d}$$

Recorrendo agora para a Tabela 16.2, podemos calcular a prevalência do nosso exemplo fictício:

$$Prevalência = \frac{15+1}{15+2+1+16} = \frac{16}{34} = 0{,}4706$$

Ou seja, a proporção de pessoas doentes no estudo foi de 47,06%.

16.10 TABELA DIAGNÓSTICO NA PRÁTICA R

Para trabalharmos os cálculos das métricas acima estudadas, podemos usar dois pacotes do R. O primeiro traz a maioria das métricas de uma forma bem objetiva, o pacote caret. O segundo pacote é o MLmetrics, cujas métricas podem ser calculadas separadamente, o que pode ser útil para calcular métricas específicas.

Vamos imaginar que um pesquisador esteja trabalhando com um marcador bioquímico revolucionário para o diagnóstico da diabetes melitos. Os dados abaixo fazem parte da coleta de informações de sua pesquisa, com o padrão ouro (o diagnóstico médico confirmado por meios clássicos) que ele chamou de PO e o novo teste que ele preferiu chamar de TESTE somente. Vamos chamar esse banco de dados de *diag*. Quando o voluntário for diagnosticado ou testado como tendo diabetes, o pesquisador o identificou com um "D" para esse indivíduo. Quando saudável, "S". Usando a função head() para ver se o *data frame* estava feito corretamente, observamos que os quatro primeiros indivíduos são diabéticos e foram testados também como diabéticos.

```
diag <- data.frame(
  PO = factor(c(rep("D", 30), rep("S", 30))),
  TESTE = factor(c(rep("D", 25),rep("S",5),rep("D", 3),rep("S",27)))
)

#ou importe o arquivo
library(rio)
diag <- import('pasta_de_arquivo/diag.txt')

head(diag)
  PO TESTE
1  D     D
2  D     D
3  D     D
4  D     D
5  D     D
6  D     D
```

No pacote caret existe uma função chamada confusionMatrix(). Esse nome vem justamente da tabela 2x2, que dependendo do contexto pode ser chamada também de matriz de confusão. Precisamos preencher a função determinando primeiramente os dados de teste (para que eles fiquem distribuídos em linha) e depois os dados do padrão ouro (para que fiquem em colunas). Não mude essa ordem ou você terá problemas interpretativos. Por fim, podemos determinar para a função qual das classes é a doença, usando o parâmetro *positive* para isso. Se você não tiver o pacote caret, lembre-se de instalá-lo.

```
install.packages('caret')
library(caret)
```

```
confusionMatrix(diag$TESTE, diag$PO, positive = "D")
Confusion Matrix and Statistics

          Reference
Prediction  D  S
         D 25  3
         S  5 27

               Accuracy : 0.8667
                 95% CI : (0.7541, 0.9406)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 2.603e-09

                  Kappa : 0.7333

 Mcnemar's Test P-Value : 0.7237

            Sensitivity : 0.8333
            Specificity : 0.9000
         Pos Pred Value : 0.8929
         Neg Pred Value : 0.8438
             Prevalence : 0.5000
         Detection Rate : 0.4167
   Detection Prevalence : 0.4667
      Balanced Accuracy : 0.8667

       'Positive' Class : D
```

Perceba que vários resultados são dispostos na tela. Primeiramente a função retorna a tabela 2x2 com as frequências absolutas. As linhas, ou *Prediction*, são os dados do TESTE, enquanto que *Reference* são os dados do padrão ouro. Perceba também que a acurácia retornada foi de 86,67% (*Accuracy*: 0.8667), a sensibilidade foi de 83,33% (*Sensitivity*: 0.8333), a especificidade foi 90,00% (*Specificity*: 0.9000), o valor preditivo positivo foi de 89,29% (*Pos Pred Value*: 0.8929), o valor preditivo negativo foi de 84,38% (*Neg Pred Value*: 0.8438) e a prevalência foi de 50,00 (*Prevalence*: 0.5000). Constate que ele confirma que o positivo para a doença foi a letra D ('Positive' Class: D).

Veja que os resultados gerados pela função, apesar de bem informativos, não retornam os valores das razões de verossimilhança positiva e negativa que estudamos acima. Para calcularmos essas duas métricas, podemos usar as funções `Sensitivity()` e `Specificity()` do pacote MLmetrics para facilitar nossa vida. Lembre-se de instalá-lo antes de abrir a biblioteca no R.

Vamos denominar os objetos com os valores das razões de verossimilhança como RVpos e RVneg. Podemos calcular diretamente da seguinte forma:

```
install.packages('MLmetrics')
library(MLmetrics)
```

```
RVpos <- Sensitivity(diag$TESTE, diag$PO)/(1 - Specificity(diag$TESTE, diag$PO))

RVneg <- (1-Sensitivity(diag$TESTE, diag$PO))/Specificity(diag$TESTE, diag$PO)
```

Verificando os resultados:

```
RVpos
[1] 5.714286

RVneg
[1] 0.1269841
```

Caso você precise calcular vários valores de razão de verossimilhança acredito que seja melhor criar as duas funções e usá-las quando necessário. Podemos fazer da seguinte maneira:

```
fRVpos <- function(teste, padraOuro){
  Sensitivity(teste, padraOuro)/(1 - Specificity(teste, padraOuro))
}

fRVneg <- function(teste, padraOuro){
  (1-Sensitivity(teste, padraOuro))/Specificity(teste, padraOuro)
}
```

Com a função fRVpos criada para calcular a razão de verossimilhança positiva e fRVneg para o cálculo da razão de verossimilhança negativa, podemos verificar os resultados usando a gora somente:

```
fRVpos(diag$TESTE, diag$PO)
[1] 5.714286

> fRVneg(diag$TESTE, diag$PO)
[1] 0.1269841
```

Observe que os resultados retornados pelo R são exatamente os mesmos, então criamos as funções corretamente.

Capítulo 4

A Curva ROC e o Estabelecimento de Pontos de Corte

Na área da saúde estabelecer pontos de corte (em inglês *cutoff*) em variáveis numéricas é muito importante, mas poucas pessoas se dão conta disso e menos ainda sabem como determinar esses limites de biomarcadores, especialmente nas dosagens bioquímicas.

Sempre que trabalhamos com uma variável numérica, seja qual for o tipo do exame, precisamos de um limite para dizer se acima ou abaixo desse limite é um sinal de alerta ou não. Por exemplo, se uma pessoa tiver uma dosagem de glicose de jejum muito superior a 99 mg/dL, há uma suspeita dessa ser diabético. No hemograma, dosagem de hemoglobina abaixo de 12 g/dL é um forte sinal de anemia.

Será que esses pontos de corte são válidos para todas as faixas etárias? Será que são válidos para ambos os sexos? Será que são válidos para todas as populações ao longo do globo terrestre? Bom, é um assunto de recorrente discussão e aperfeiçoamento na área da saúde.

Existem diversos métodos para chegarmos a um ponto de corte de uma variável numérica. Pode existir uma comunidade de especialistas em uma determinada doença ou condição que afirme que um ponto de corte tal é importante para diferir doentes e não doentes, por exemplo. Sendo dessa forma, é considerado a expertise do(s) especialista(s), mas não é nosso objetivo aqui falarmos sobre uma doença especificamente. E mesmo assim, os especialistas precisam de um ponto de partida.

Aqui, aprenderemos um passo simples que atualmente tem sido muito utilizado, usando-se a curva ROC, analisando sua área sob a curva e estipulando o índice de Youden, que nos trará o ponto de corte em si. Esses passos requerem o mínimo de conhecimento prévio do capítulo anterior, especialmente de sensibilidade e especificidade, pois o equilíbrio entre ambos é fundamental para acharmos um bom e expressivo ponto de corte.

Obviamente, quando dividimos uma variável numérica em duas partes, estamos também categorizando ela, ou seja, acima de X e abaixo de X. Dessa forma, é possível usarmos essa estratégia para construirmos tabelas de frequência também.

17.1 A CURVA ROC

A curva ROC (*Receiver Operating Characteristic*) foi desenvolvida entre os anos de 1950 e 1960 para avaliar a detecção de sinais de radar e também na psicologia sensorial. Verdade seja

dita, o ramo da psicologia tem mais estatística do que muitos possam imaginar. Existem psicólogos especializados em psicometria, que são verdadeiros artistas da bioestatística.

Antes que a curva ROC fosse efetivamente criada, Louis Leon Thurstone (1887-1955) trabalhou com análises discriminantes. Ele demonstrou que métodos psicanalíticos poderiam ser quantificados, ter uma medição físico-matemática. O modelo matemático de Thrustone vislumbra a sobreposição das distribuições de intensidade psicológica de dois estímulos semelhantes.

Nesse modelo (Figura 17.1), assume em uma distribuição normal o efeito sensorial quando não há nenhum estímulo (*So*), que alguns pesquisadores da área chamam de ruído. A distribuição da direita ocorre quando há estímulo (*Si*), então, houve um sinal. Existe um ponto chamado de *c* que foi estipulado por Thrustone o qual a partir deste haveria uma resposta positiva do estímulo. Esse ponto é a interseção entre as distribuições.

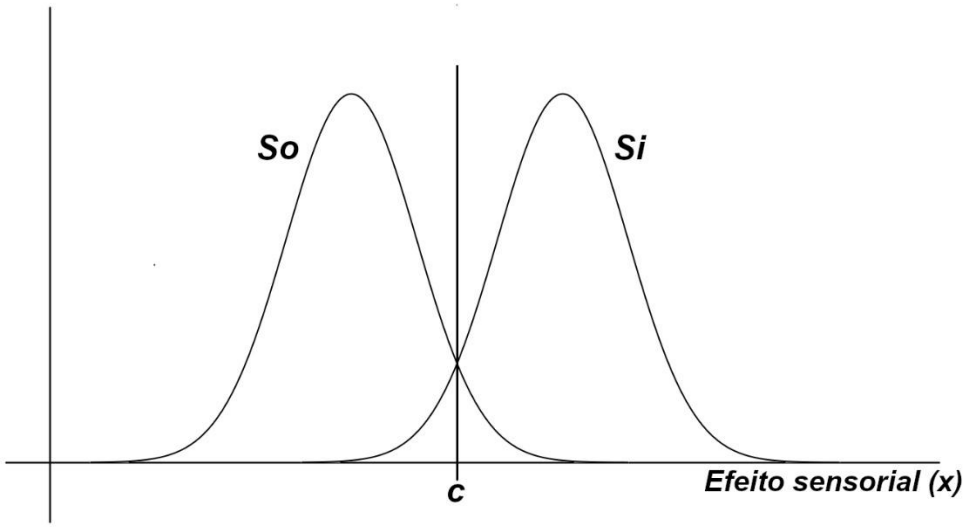


Figura 17.1. Modelo genérico de reconhecimento de Thrustone para representação de detecção sensorial.

Esse modelo pode não ser o mais adequado em todas a situações, mas ele dá uma ideia fundamental ao critério de decisão para considerar um ponto de corte: os valores achados de um biomarcador quando uma doença está presente, precisa ser diferente em relação aos valores quando não há doença (ruído). Existirá, a prática nos mostra isso, valores de biomarcadores considerados normais quando uma doença está presente, bem como valores alterados desse biomarcador quando a doença está ausente. Contudo, para que seja um bom biomarcador, essas situações precisão ser raras. Estamos falando da região próxima à interseção das distribuições.

Pois bem, nesse intermeio, já sabemos que um teste diagnóstico com alta sensibilidade e alta especificidade é o que qualquer profissional da área da saúde possa buscar. Contudo, já vimos que essa relação sensibilidade e especificidade geralmente não é perfeita. Por vezes, aumentamos a sensibilidade do teste, porém a especificidade cai, e vice-versa.

A curva ROC é um tipo de análise que gera diversos pontos de corte, sendo que cada um deles é estabelecido um par sensibilidade e especificidade do teste diagnóstico. Dessa forma, podemos verificar várias possibilidades de *cutoff points* e equilibrar a relação sensibilidade e especificidade.

A curva ROC, na realidade, é um diagrama da sensibilidade (verdadeiros positivos, no eixo y) em função da proporção de falsos positivos (1 – especificidade, no eixo x) para cada ponto de corte de uma variável contínua, a qual será dicotomizada de forma sequenciada. Nesta conformação gráfica (sensibilidade vs. 1 - especificidade) ambos os valores dos eixos são representados de forma crescente. Entretanto, podemos construir gráficos usando no eixo x a especificidade puramente, contudo, os valores estarão na ordem decrescente. Na verdade, nada muda a não ser a forma visual dos valores do eixo x.

A Figura 17.2 representa uma demonstração da curva ROC. Primeiramente, percebemos que no eixo y estão os valores de sensibilidade na forma decimal. No eixo x está a especificidade, nesse caso com seus valores na forma decrescente e também na forma decimal. Esses valores poderiam ser demonstrados na forma percentual também; a conformação do gráfico seria a mesma. Ao centro temos uma reta mais sutil de cor cinza na diagonal do gráfico, que é uma linha de referência quando a área sob a curva tem o valor 0,5. Veremos mais adiante o que isso significa, mas diante mão entenda que quanto mais longe dessa linha a curva ROC estiver, melhor será o nível de distinção entre doentes e não doentes através da variável numérica. Por fim, a curva ROC em si, em preto mais evidente, que é arqueada e com várias com várias curvas.

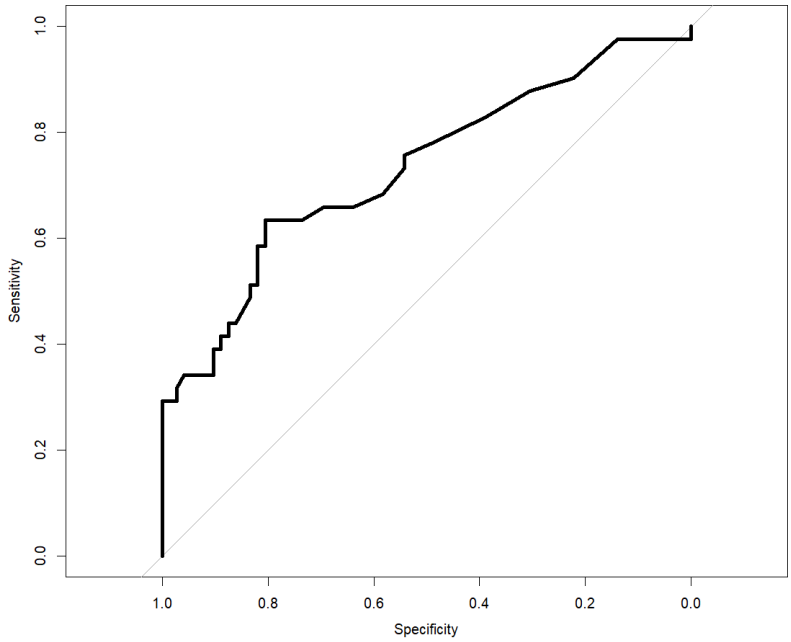


Figura 17.2. Exemplo de uma curva ROC.

Além da sensibilidade, especificidade, na própria ROC há uma métrica a mais nesse contexto, que chamamos de área sob a curva (ou do inglês *Area Under The Curve*; AUC). O nome é praticamente autoexplicativo, ou seja, a curva ROC forma um polígono no gráfico quando consideramos toda a área que há da ROC até tocar no eixo x. Esse polígono está presente em cinza na Figura 17.3.

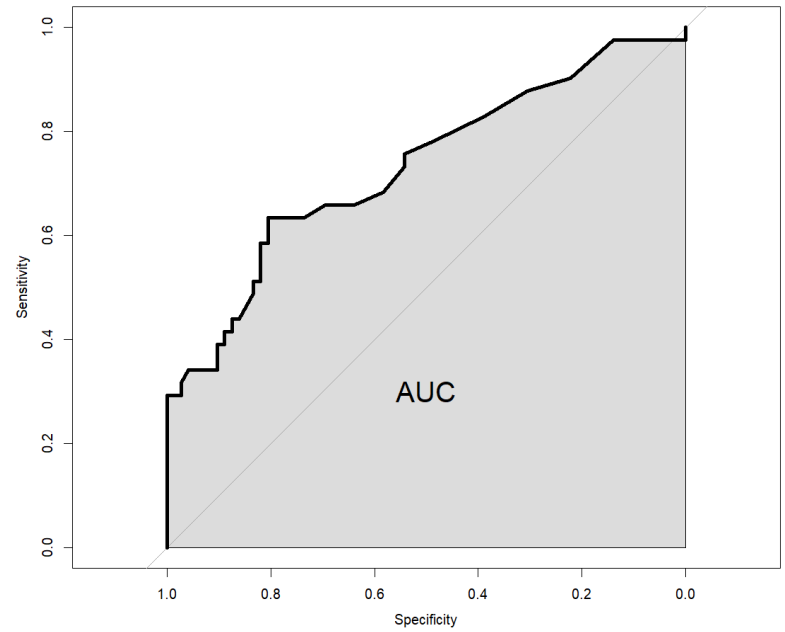


Figura 17.3. A área sob a curva ROC.

A AUC resume a capacidade do teste de distinguir entre as classes positiva e negativa. Essa métrica varia de 0,5 a 1, onde 0,5 indica um teste aleatório e 1 representa um teste perfeito, com erro de classificação zero. Quanto maior a AUC, melhor o desempenho do teste, indicando sua capacidade de separar corretamente as classes. Graficamente esses extremos podem ser representados respectivamente nos gráficos A e B da Figura 17.4.

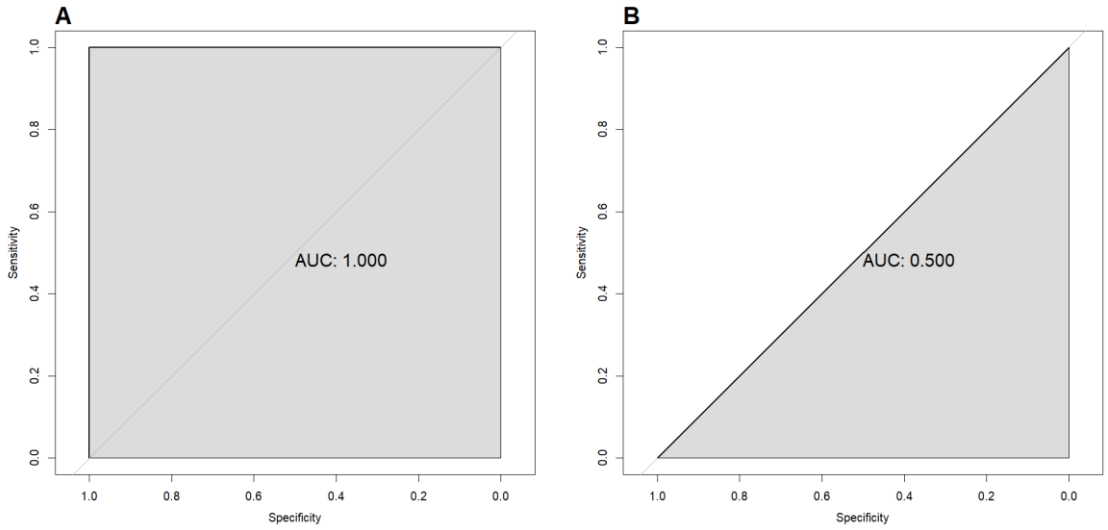


Figura 17.4. Diferença gráfica entre as áreas sob a curva quando iguais a 1,0 e 0,5, respectivamente.

De forma interpretativa, Akobeng (2007) relaciona os valores de AUC com as acurácias de teste, destrinchando melhor o caráter classificatório dos valores das AUC. Iremos transportar suas ideias para a Tabela 17.1.

Tabela 17.1. Relação da AUC e acurácias.

Valor da AUC	Interpretação
1,0	Predição perfeita
0,9 – 1,0	Alta acurácia
0,7 – 0,9	Acurácia moderada
0,5 – 0,7	Baixa acurácia
0,5	Resultado casual

Fonte: Akobeng (2007), modificada.

A Figura 17.5 uni o entendimento que obtivemos com o modelo genérico de Thrustone com três diferentes curvas ROC. A ROC em azul (curva 1) é a situação perfeita cujos valores da variável numérica (um biomarcador quantitativo) não surgem dúvidas quanto a quem está doente e sadio. AUC igual a 1.

A ROC em vermelho (curva 2) é uma situação mais parecida com a realidade, na maioria dos casos. Existirá valores que podem gerar confusão, mesmo que poucos, os quais estarão em um patamar que não saberemos com precisão se a pessoa está doente ou sadia. Estamos falando da região *c* da Figura 17.1. Vamos destrinchar mais essa ROC, perceba que há três pontos: A, B e C. São possíveis pontos de corte. No ponto A, observamos uma sensibilidade de 50% e uma especificidade de 90% (lembre-se que pode estar no gráfico 1 – especificidade). Aparenta uma ótima especificidade, mas a sensibilidade é relativamente baixa. No ponto B, a sensibilidade é aproximadamente 80% e a especificidade de 75%, aproximadamente. Apesar da redução da especificidade, conseguimos nesse ponto equilibrar melhor as probabilidades de surgimento de falsos positivos e falsos negativos. No ponto C, temos uma sensibilidade alta, de 90%, mas a especificidade ronda nos 40%. Voltamos a desequilibrar as probabilidades dos erros.

A curva em laranja da Figura 17.5-A demonstra a situação em que a ROC se iguala à linha de referência. Perceba na Figura 17.4-B, que as distribuições da curva 3 são semelhantes e se sobrepõem bastante. Não há uma maneira adequada de distinguir doentes e não doentes através dessa variável quantitativa que compõe a curva laranja.

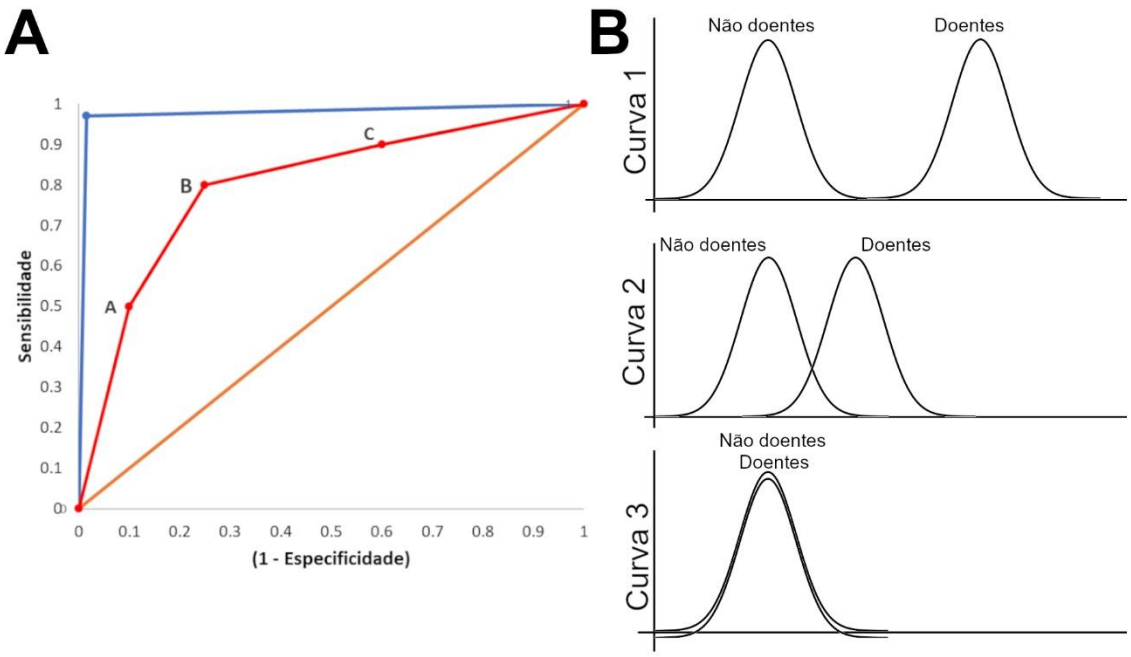


Figura 17.5. Curvas ROC (A) e modelos de Thrustone (B).
Curva ROC 1 (em azul); curva ROC 2 (em vermelho); curva ROC 3 (em laranja).

Através dessas diferenças e avaliando tais métricas, podemos estipular pontos de corte adequados para a nossa realidade, além de comparar curvas ROC de biomarcadores diferentes e verificar qual é o melhor para distinguir indivíduos saudáveis de não saudáveis, por exemplo.

Vamos praticar no R como construir e calcular as ROC e AUC respectivas. Para isso, utilizaremos o pacote pROC, até então, o melhor no assunto. Se você ainda não tiver, instale-o. Usaremos também um banco de dados pertencente a esse pacote, chamado de aSAH (vem do inglês *aneurysmal subarachnoid haemorrhage* ou hemorragia subaracnoidea aneurismática). Esses dados fazem parte de um estudo que avalia o risco na evolução pós-SAH (bom ou ruim / *Good* ou *Poor*), pois caso seja ruim esses pacientes requerem manejo de saúde específico. Para essa avaliação, foi usado um biomarcador quantificado através do plasma sanguíneo chamado S100B. É uma proteína frequentemente encontrada em astrócitos e células de Schwann no cérebro de humanos e outros animais. Quando há lesão do sistema nervoso central, S100B é liberada para a corrente sanguínea e podemos quantificar isso por um exame de sangue. Portanto, o aumento plasmático da concentração de S100B sérico pode sugerir lesão glial ou astrogliose reativa.

Em suma, nessa avaliação, as duas variáveis fundamentais serão "outcome" que é o nosso desfecho (prognóstico bom ou ruim) e a dosagem sérica de S100B em ng/mL, variável numérica "s100b". No banco há outras variáveis, mas não vamos nos preocupar com elas nesse momento. Vamos abrir e ver os dados.

```
install.packages('pROC')
library(pROC)
data(aSAH)

head(aSAH)
   gos6 outcome gender age wfns s100b  ndka
29    5    Good Female  42    1  0.13  3.01
30    5    Good Female  37    1  0.14  8.54
31    5    Good Female  42    1  0.10  8.09
32    5    Good Female  27    1  0.04 10.42
33    1    Poor Female  42    3  0.13 17.40
34    1    Poor   Male  48    2  0.10 12.75
```

Para construirmos a ROC e calcularmos a AUC usaremos a função roc() do pacote pROC. Devemos determinar alguns parâmetros importantes nessa função, como o nome do banco de dados, a variável desfecho (categórica), a variável numérica e estabelecer como TRUE o parâmetro "plot" para que o gráfico seja demonstrado. As duas variáveis podem ser acrescentadas separadamente, mas aqui demonstrarei a sintaxe no formato de fórmula. Ficará assim:

```
roc(outcome ~ s100b, aSAH, plot = TRUE)
```

```
Setting levels: control = Good, case = Poor
Setting direction: controls < cases

Call:
roc.formula(formula = outcome ~ s100b, data = aSAH, plot = TRUE)

Data: s100b in 72 controls (outcome Good) < 41 cases (outcome Poor).
Area under the curve: 0.7314
```

Após rodar a função, o que o R retorna será primeiramente um ponto de checagem para gente. Em *Setting levels* ele demonstra qual classe foi considerado controle e qual foi considerada caso. Está tudo correto em nosso exemplo. Por fim, o que temos de importante é a AUC, cujo valor retornado foi 0,7314. Uma acurácia moderada. Ainda mais, obtivemos a criação da curva ROC, a qual está expressa na Figura 17.6.

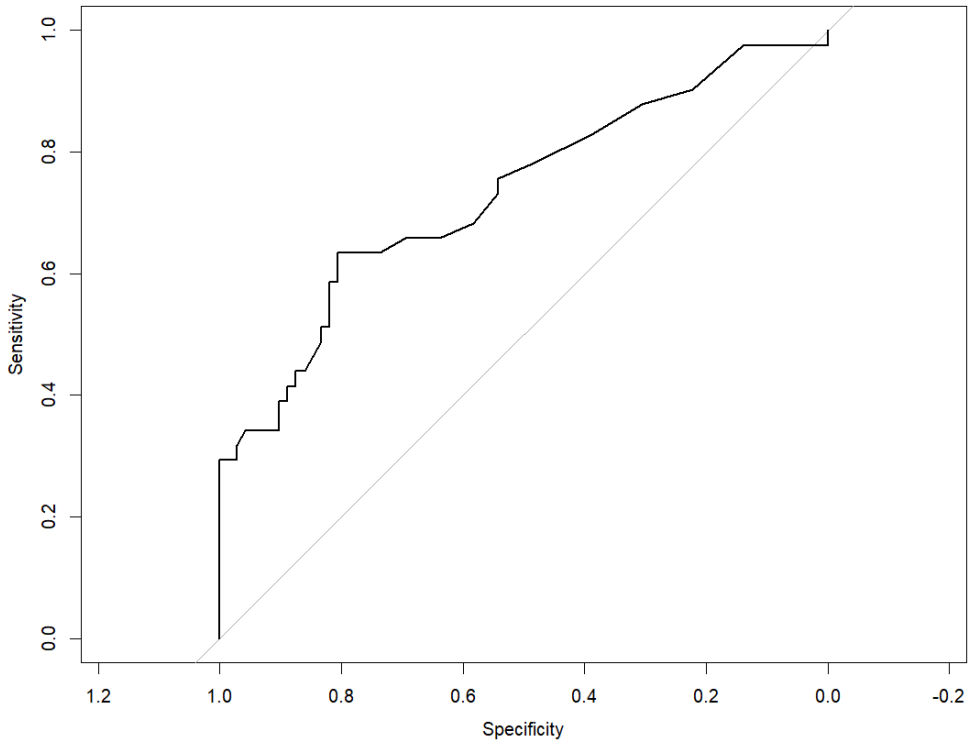


Figura 17.6. Curva ROC do prognóstico da hemorragia subaracnoidea aneurismática segundo a S100B em ng/mL.

A função roc() calculará tudo necessário. Podemos plotar separadamente o gráfico usando a função plot(), que também nos possibilita adicionar o valor da AUC no gráfico da ROC. Para isso, precisamos determinar o parâmetro "print.auc" como TRUE. Para aumentar o tamanho da fonte do valor de AUC, modifique o valor do parâmetro numérico "print.auc.cex". Se você desejar hachurar a área sob a curva, o que é opcional e poucos pesquisadores o fazem, determine o

parâmetro "auc.polygon" como TRUE. Para aumentar ou diminuir a espessura da linha da ROC, modifique o parâmetro numérico "lwd". O gráfico retornado será como a Figura 17.7. Para aprender mais parâmetros que configure o gráfico da curva ROC, acesse a documentação do pacote pROC, especificamente na função plot.roc(), que é justamente uma alternativa à plot().

```
#Figura 17.7
  #Forma direta
roc(outcome ~ s100b, aSAH, plot = T,
    lwd = 4,
    print.auc = T,
    auc.polygon = T,
    print.auc.cex = 1.5)

  #Forma indireta
result_roc <- roc(outcome ~ s100b, aSAH)
plot(result_roc,
     lwd = 4,
     print.auc = T,
     auc.polygon = T,
     print.auc.cex=1.5)
```

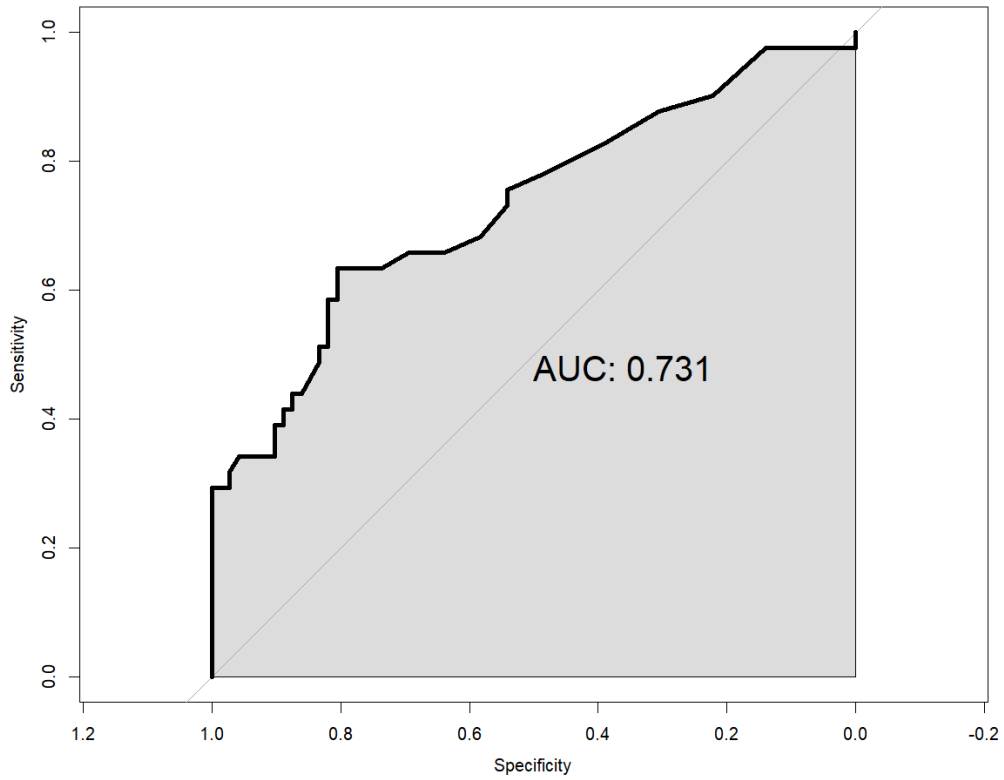


Figura 17.7. Curva ROC do prognóstico da hemorragia subaracnoidea aneurismática segundo a S100B, hachurada e com o valor de AUC.

17.2 COMPARANDO CURVAS ROC DIFERENTES

Como observado, a curva ROC é uma ferramenta que nos retornará métricas adicionais à questão da qualidade preditiva de um teste diagnóstico usando uma variável numérica. No exemplo acima, trabalhamos uma variável que era uma dosagem sérica de um biomarcador: S100B. A pergunta que fica é: será que para avaliarmos o nosso desfecho em questão, esse é o melhor biomarcador? Será que podemos comparar biomarcadores diferentes?

A resposta é sim. Para o mesmo desfecho, podemos comparar curvas ROC diferentes e avaliar se uma é melhor que a outro, ou ainda se são exames que podem se complementar (o que normalmente a gente faz na área da saúde).

Para isso, precisamos construir duas ROC diferentes usando a função `roc()` do pacote pROC. Depois, devemos testar a igualdade entre elas usando a função `roc.test()` do mesmo pacote. Basta especificar nesta função os dois objetos que criaremos a partir de `roc()`.

Vamos verificar se a idade (*age*) é uma variável interessante para avaliarmos o prognóstico da hemorragia subaracnoidea aneurismática e comparar com a dosagem de S100B.

```
library(pROC)
data(aSAH)

#ROC usando a dosagem de S100B
roc1 <- roc(aSAH$outcome, aSAH$s100b)
Setting levels: control = Good, case = Poor
Setting direction: controls < cases

#ROC usando a idade
roc2 <- roc(aSAH$outcome, aSAH$age)
Setting levels: control = Good, case = Poor
Setting direction: controls < cases

roc.test(roc1, roc2)

      DeLong's test for two correlated ROC curves

data:  roc1 and roc2
Z = 1.7604, p-value = 0.07835
alternative hypothesis: true difference in AUC is not equal to 0
95 percent confidence interval:
 -0.01319368  0.24591726
sample estimates:
AUC of roc1 AUC of roc2
  0.7313686   0.6150068
```

O R retornará resultados do teste de DeLong, que não aprofundaremos no seu conceito matemático por fugir do escopo do livro, porém, basicamente o teste envolve a correlação entre as

classificações das duas ROC, o que é calculado a partir da matriz de covariância das AUC. Através desse teste de comparação, um p-valor é gerado. Se ele for menor ou igual a 0,05, podemos afirmar que as duas curvas ROC são estatisticamente diferentes. Caso contrário, as curvas são estatisticamente iguais (obviamente considerando o nível de significância de 5%). Além disso, é demonstrado nos resultados o valor das duas AUC: 0.73 e 0.61 (respectivamente, roc1 e roc2).

Pois bem, em nosso exemplo prático, o p-valor retornado foi de 0,07835. Então, tecnicamente as ROC se equivalem. Mas, vamos verificar como ficaram visualmente as curvas para cada variável numérica e suas respectivas AUC no gráfico. Pra isso, vamos estabelecer TRUE para os parâmetros "plot" e "print.auc" para a primeira ROC, e adicionar a curva roc2 no gráfico já feito estabelecendo TRUE para o parâmetro "add" da segunda ROC, e vamos mudar sua cor para vermelho modificando o parâmetro "col". Mudando o parâmetro "print.auc.col" podemos mudar a cor do valor da AUC. Com "print.auc.adj" podemos ajustar seu local no gráfico. Veja como fica o código:

```
#Figura 17.8
  #Forma direta de adição de duas ROC e suas AUC
roc(aSAH$outcome, aSAH$s100b, plot = T, print.auc = T,
    lwd = 4, print.auc.cex = 2)
roc(aSAH$outcome, aSAH$age, plot = T, add = T,
    col = 'red', lwd = 4, print.auc = T,
    print.auc.cex = 2, print.auc.col = 'red', print.auc.adj = c(0,5))
```

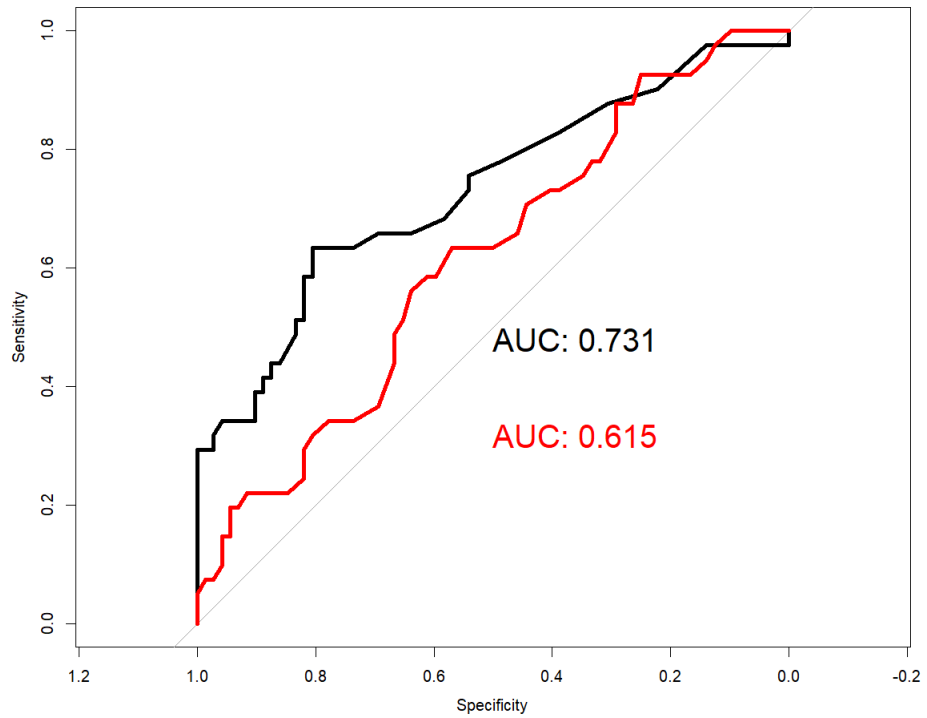


Figura 17.8. Comparando duas curvas ROC.
Nota: Em preto a dosagem de S100B e em vermelho a idade.

Já vimos que as duas ROC são estatisticamente similares. Podemos verificar visualmente que as ROC são parecidas, a ponto que as porções superiores de ambas se entrelaçam. Contudo, perceba que de forma geral, a ROC da idade fica mais próxima da linha de referência do que a da dosagem de S100B. Daí seu valor de AUC ser menor. Mas não são grandes diferenças (em nosso exemplo).

17.3 ESTIMANDO O PONTO DE CORTE IDEAL

Até aqui, falamos que a curva ROC estabelece diversos pontos de corte, vinculando a cada um deles suas respectivas sensibilidade e especificidade. Contudo ainda não demonstramos esse tipo de resultado. Para isso, vamos conferir os resultados da nossa primeira ROC, sobre o marcador S100B. Vamos salvar em um objeto que chamaremos de "result_roc" e explorar o que está gravado dentro dele como parte dos resultados. Vamos salvar em um segundo objeto do tipo *data frame* como LSE e visualizar dos resultados dos limites (pontos de corte sugeridos em toda a sua extensão dos dados), a sensibilidade e a especificidade de cada limite. Com head(,10) vamos expor as 10 primeiras linhas de LSE. Vejamos:

```
library(pROC)
data("aSAH")

result_roc <- roc(outcome ~ s100b, aSAH)

LSE <- data.frame(
  limites = result_roc$thresholds,
  sensibilidades = result_roc$sensitivities,
  especificidades = result_roc$specificities)

head(LSE, 10)
   limites sensibilidades especificidades
1     -Inf      1.0000000      0.00000000
2    0.035      0.9756098      0.00000000
3    0.045      0.9756098      0.06944444
4    0.055      0.9756098      0.11111111
5    0.065      0.9756098      0.13888889
6    0.075      0.9024390      0.22222222
7    0.085      0.8780488      0.30555556
8    0.095      0.8292683      0.38888889
9    0.105      0.7804878      0.48611111
10   0.115      0.7560976      0.54166667
```

A questão final agora é: qual dentre todos esses pontos de corte seria o melhor frente ao meu desfecho?

Para responder isso, existem diversas técnicas. Aprenderemos aqui sobre o Índice de Youden. Em exatos 1950, na revista científica chamada Cancer, Youden publicou um artigo chamado Índice para Classificação de Testes de Diagnóstico (*Index for rating diagnostic tests*). Com ele, o autor ensina a calcular o seu índice *J*.

Baseado na tabela 2x2 de diagnóstico (Tabela 16.1), ele estabeleceu que:

$$J = \left[\frac{a}{a+c} + \frac{d}{b+d}\right] - 1 \quad \text{ou} \quad J = sensibilidade + especificidade - 1$$

Esse índice pode variar de -1 a +1, sendo que +1 seria o teste diagnóstico perfeito e -1 seria a pior das hipóteses, pois o teste erraria exatamente todas as vezes. O valor 0 indica que o teste não é melhor do que adivinhar ao acaso (é jogar uma moeda para cima e tentar adivinhar quem está doente e quem não está).

Mas como o índice escolhe o melhor ponto de corte? Pois bem, na curva ROC há diversos pontos possíveis, cada um com as suas respectivas sensibilidades e especificidades, como já foi dito e demonstrado. Mas, perceba que com isso podemos calcular um índice *J* para cada ponto de corte sugerido pela ROC. O maior valor de *J* entre todos os pontos estabelecidos será ali o nosso ponto de corte da variável numérica, sugerido por Youden.

Para calcularmos diretamente a ROC e os valores de *J* e o melhor ponto de corte da variável numérica, usaremos o pacote OptimalCutpoints. Não é o único pacote do R disponibilizado para esse fim, mas é o que me agrada mais. Caso não lhe satisfaça, teste outros disponíveis no site oficial de distribuição do R.

Nesse pacote há um banco de dados chamado "elas". Ele foi obtido no Departamento de Cardiologia do Hospital Geral da Galiza (Santiago de Compostela, Espanha). Este estudo avaliou a utilidade clínica da enzima elastase leucocitária no diagnóstico de doença arterial coronariana (Amaro, 1995). Há três variáveis nesse conjunto de dados: "elas" (dosagem de elastase sérica dada em µg/L), "status" (0 = controles e 1 = doentes), e "gender" (gênero do indivíduo: *male* ou *female*).

Vamos instalar o pacote e abrir a biblioteca. Não é obrigado a mudar os dados do banco original, mas para melhor compreensão, vamos atribuir um valor do tipo *string* aos valores numéricos da variável status. Vamos sobrescrever os valores 0 e 1 para controle e caso, respectivamente.

```
install.packages('OptimalCutpoints')
library(OptimalCutpoints)

data(elas) #abrindo o banco
```

```
#modificando status
elas$status <- ifelse(elas$status == 0, 'controle', 'caso')

head(elas)
  elas   status gender
1   27     caso   Male
2   34     caso   Male
3   41     caso   Male
4   20 controle Female
5   25     caso Female
6   26     caso   Male
```

Para estabelecer o melhor ponto de corte para a dosagem de elastase sérica, usaremos a função optimal.cutpoints(). Nela há alguns parâmetros a serem preenchidos com seus respectivos argumentos. Primeiramente devemos dizer qual é o banco de dados iremos utilizar no parâmetro "data"; devemos pôr a variável numérica no parâmetro "X" (atenção: x maiúsculo); devemos pôr o desfecho no parâmetro "status"; no parâmetro "tag.healthy" colocaremos a classe de indivíduos saudáveis ou controles; em "methods" devemos pôr o método que será usado para estimar o ponto de corte, que em nosso caso será "Youden".

Para analisar os resultados vamos usar a função summary().

```
pontoCorte <- optimal.cutpoints(data = elas,
                                X = "elas",
                                status = "status",
                                tag.healthy = 'controle',
                                methods = "Youden")
summary(pontoCorte)

Call:
optimal.cutpoints.default(X = "elas", status = "status", tag.healthy = "controle",
    methods = "Youden", data = elas)

Area under the ROC curve (AUC):  0.744 (0.659, 0.828)

CRITERION: Youden
Number of optimal cutoffs: 1

                   Estimate
cutoff           37.0000000
Se                0.6875000
Sp                0.6666667
PPV               0.8148148
NPV               0.5000000
DLR.Positive      2.0625000
DLR.Negative      0.4687500
FP               15.0000000
FN               30.0000000
Optimal criterion  0.3541667
```

Como resultado, vemos que a AUC e seu intervalo de confiança foram aproximadamente 0,74 (0,66-0,83). O ponto de corte está descrito em *cutoff*, que foi 37,00 µg/L. Nesse ponto de corte a sensibilidade é de aproximadamente 68,75% e especificidade de 66,67%.

Vamos avaliar a ROC dessa análise, construir o data frame de resultados e verificar mais ou menos a linha entre os limite (pontos de cortes sugeridos) que vieram da função roc() do pacote pROC.

```
library(pROC)
result_roc <- roc(status ~ elas, elas)

LSE <- data.frame(
  limites = result_roc$thresholds,
  sensibilidades = result_roc$sensitivities,
  especificidades = result_roc$specificities)

LSE[35:45,]
   limites sensibilidades especificidades
35   40.50      0.7333333       0.5416667
36   39.75      0.7333333       0.5937500
37   39.25      0.7111111       0.5937500
38   38.50      0.6666667       0.6458333
39   37.50      0.6666667       0.6770833
40   36.50      0.6666667       0.6875000
41   35.50      0.6000000       0.6979167
42   34.50      0.5555556       0.6979167
43   33.50      0.5555556       0.7291667
44   32.50      0.5333333       0.7395833
45   31.50      0.5111111       0.7604167
```

Foi possível perceber que o método de Youden sugeriu um ponto de corte entre as linhas 39 e 40. Perceba que as sensibilidades e especificidades dessas linhas estão próximas a sugerida pela função optimal.cutpoints(). É um ponto de equilíbrio para não ter ótima sensibilidade e péssima especificidade, ou vice-versa.

Podemos plotar a curva ROC com destaque ao ponto de corte sugerido pelo método de Youden, usando a função plot(). O R retornará a Figura 17.9. Nela você observará 1 – especificidade e a sensibilidade do ponto de corte sugerido, também a AUC e seu intervalo de confiança. Além da curva ROC a função retorna também a curva PROC que é o complementar do valor preditivo negativo em função do valor preditivo positivo. O gráfico PROC não é muito abordado na bioestatística, então não vamos nos aprofundar nele.

```
#Figura 17.9
plot(pontoCorte)
```

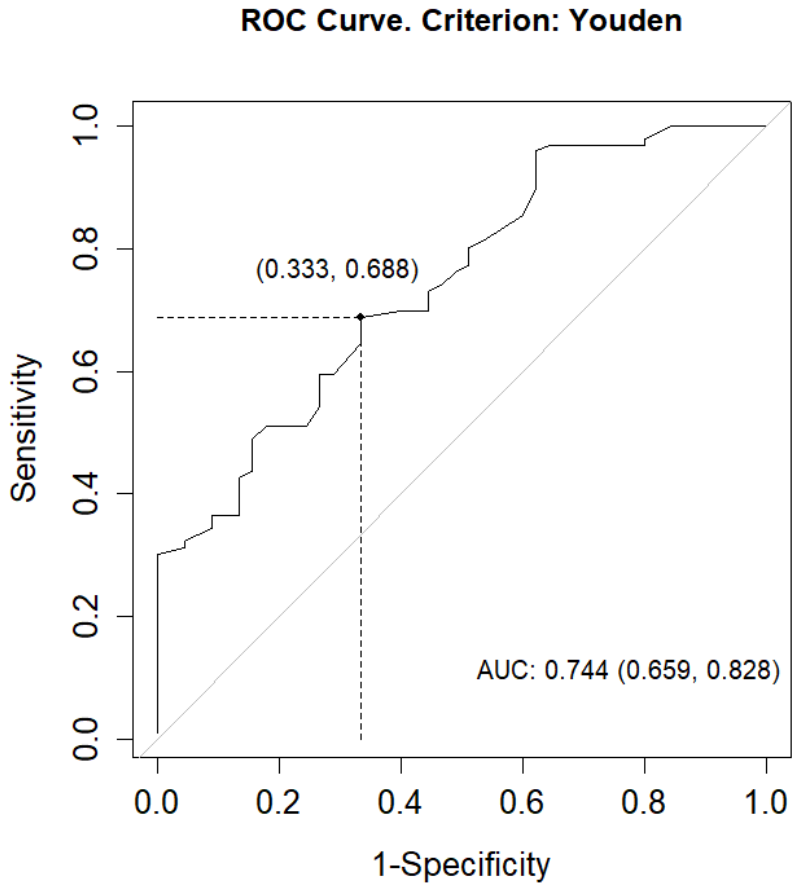


Figura 17.9. Curva ROC para estabelecer o ponto de corte usando o método de Youden.

O que temos até aqui é o ponto de corte entre casos e controles, que foi de 37,00 µg/L. Mas a métricas de AUC, sensibilidade e especificidade não foram tão satisfatórias assim, concorda? Vamos nos aprofundar nesse caso. Vamos visualizar em um gráfico de densidade (aprenderemos melhor mais à frente) como estão as distribuições dos valores de elastase entre casos e controles. O comando abaixo gerará a Figura17.10.

Um argumento importante que iremos pôr na figura é justamente o ponto de corte que achamos. Ele está "escondido" dentro do objeto pontoCorte que criamos acima. Depois de alguns cifrões, teremos o valor exato. Vamos demonstrar esse valor usando uma reta vertical tracejada dentro do gráfico para usamos como referência, lembrando que esse é o ponto de corte que diferencia os casos dos controles.

```
#Figura 17.9
library(tidyverse)
elas %>%
```

```
  ggplot(aes(x = elas, fill = status)) +
  geom_density(alpha = 0.4) +
  scale_fill_manual(values = c('red', 'green')) +
  geom_vline(xintercept = pontoCorte$Youden$Global$optimal.cutoff$cutoff,
             linetype = "dashed") +
theme_classic()
```

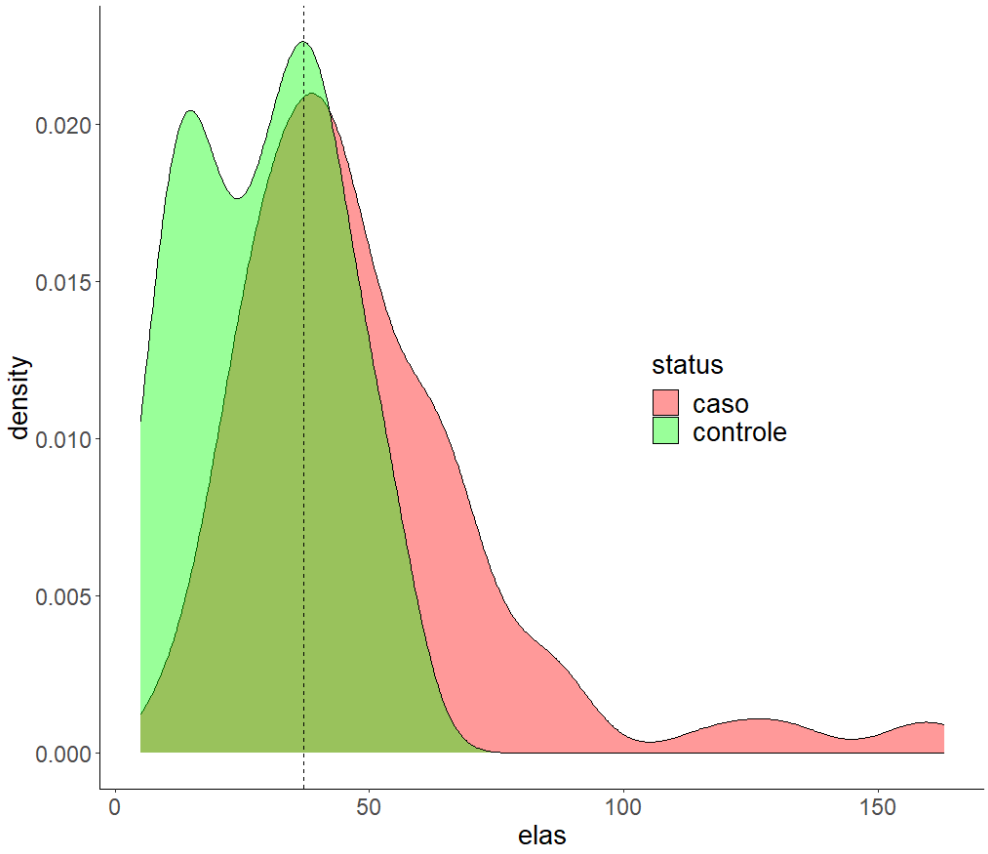


Figura 17.10. Gráfico de densidade das distribuições de elastase entre casos e controles.

Percebemos na Figura 17.10 que o ponto de corte se encontra justamente na sobreposição dos maiores picos das duas densidades. Em verde, observamos que no grupo controle há dois picos significativos, o que não é muito normal em uma população homogênea. No grupo caso há um pico significo e outro pico menos expressivo depois de 100 µg/L.

Esse tipo de comportamento pode surgir quando a nossa população tem intrinsicamente algum fator coadjuvante (ou conjunto de fatores) que interferem nas dosagens de elastase. Fatores comuns dessas situações podem ser a idade, o sexo, a etnia, comorbidades, enfim. No nosso exemplo, talvez o sexo interfira nessas dosagens, pois homens e mulheres podem apresentar taxas sanguíneas de elastase diferentes.

Vamos testar o sexo como covariável da nossa análise e verificar se esse fator influencia no ponto de corte da dosagem de elastase. Para isso, devemos pôr a variável *gender* como argumento no parâmetro "categorical.cov". Como temos masculino (*male*) e feminino (*female*), haverá uma saída como dois pontos de corte, um para cada sexo. Vejamos:

```
pontoCorte2 <- optimal.cutpoints(data = elas,
                                 X = "elas",
                                 status = "status",
                                 tag.healthy = 'controle',
                                 methods = "Youden",
                                 categorical.cov = 'gender')

summary(pontoCorte2)

Call:
optimal.cutpoints.default(X = "elas", status = "status", tag.healthy = "controle",
    methods = "Youden", data = elas, categorical.cov = "gender")

**********************************************
Female
**********************************************

Area under the ROC curve (AUC):  0.818 (0.684, 0.952)

CRITERION: Youden
Number of optimal cutoffs: 1

                    Estimate
cutoff            46.0000000
Se                 0.6666667
Sp                 0.8181818
PPV                0.7142857
NPV                0.7826087
DLR.Positive       3.6666667
DLR.Negative       0.4074074
FP                 4.0000000
FN                 5.0000000
Optimal criterion  0.4848485


**********************************************
Male
**********************************************

Area under the ROC curve (AUC):  0.722 (0.612, 0.831)

CRITERION: Youden
Number of optimal cutoffs: 1

                    Estimate
cutoff            38.0000000
Se                 0.6543210
Sp                 0.7391304
```

```
PPV                0.8983051
NPV                0.3777778
DLR.Positive       2.5082305
DLR.Negative       0.4676834
FP                 6.0000000
FN                28.0000000
Optimal criterion  0.3934514
```

Interpretando os resultados, vemos primeiramente que o R retornou os dois conjuntos de resultados, um para as mulheres e outro para os homens. Isso é um *checkpoint* importante para sabermos se identificamos corretamente a covariável. Para o conjunto de resultados feminino, verificamos que o ponto de corte ideal é 46,0 µg/L (maior do que quando estabelecido sem o uso de covariáveis). A sensibilidade nesse ponto foi de 66,67% e a especificidade foi de 81,81% e a AUC desta ROC foi 0.818. No grupo dos homens, o ponto de corte sugerido foi de 38,0 µg/L, com sensibilidade de 65,43% e especificidade de 73,91%, com AUC de 0.722.

Em termos gerais, melhoramos as métricas, principalmente de especificidade. Então, tratando da elastase sérica é melhor definir pontos de corte para cada sexo. Esse tipo de comportamento ocorre com outros biomarcadores séricos como a hemoglobina, contagem de hemácias, hematócrito, velocidade de hemossedimentação, dosagem de ácido úrico, creatinina, ferritina, entre vários outros. Então, é preciso testar diversas hipóteses antes de fechar o ponto de corte definitivo.

O parâmetro “methods” da função optimal.cutpoints() aceita diversos métodos possíveis. O de Youden é um entre dezenas. Se a sua pesquisa trata justamente de estabelecer pontos de contes em variáveis contínuas, dê uma olhada aprofundada na documentação do pacote OptimalCutpoints.

Capítulo 81

Tópicos Fundamentais em Epidemiologia

Conceitualmente, a epidemiologia é o estudo científico dos determinantes e das condições de ocorrência e agravos das doenças humanas. É uma disciplina bastante abrangente, uma vez que envolve coleta, análise e interpretação de dados para fins de compreensão das mazelas que se espalham, além de verificar os fatores que influenciam o desenvolvimento de doenças, para planejamento *a posteriori* de medidas de prevenção e controle.

Etimologicamente falando, epidemiologia (epi = sobre; demo = população; logos = estudo) significa o estudo sobre populações (PEREIRA, 2002). Ela desempenha um papel crucial na saúde pública, fornecendo informações que servem para orientar políticas públicas de saúde, suas intervenções e práticas de assistência médica, por exemplo.

Não é à toa que este livro trata dos fundamentos básicos dessas duas disciplinas, bioestatística e epidemiologia, porque de fato há uma interseção de ideias e métodos usados entre ambos. A epidemiologia não existe sem um tanto de bioestatística. Ela mexe com frequências constantemente, coisa a qual é fundamental na parte de estatística descritiva.

Ao longo deste capítulo, vamos passar pelos conceitos epidemiológicos fundamentais. Pontos importantes para que qualquer pessoa estudiosa da área biológica entenda minimamente como as doenças evoluem.

Nesse quesito já teremos um conceito interessante, que é sobre evolução (no sentido da progressão da doença). Segundo o dicionário online Dicio (www.dicio.com.br), um dos conceitos de evolução é: “Progresso; processo em que há modificação constante e progressiva, alterando um estado ou uma condição”. A evolução pode ser benéfica ou maléfica. Em hospitais ouvimos bastante: o paciente evoluiu a óbito. Para quem tem na mente que evoluir é sempre bom, talvez se engasgue nessa frase, porque o óbito não é a condição que a maioria da população gostaria de ter. Então, quando falamos em evolução, nesse contexto, queremos informar a sequência de fatos progressivos até desencadear em uma cura, um óbito ou estabilidade da doença.

18.1 ACASO, CAUSA E EFEITO

Boa parte do que vimos aqui na bioestatística se refere a possíveis relações entre variáveis e o desfecho (a doença, por exemplo). Intrinsicamente, boa parte das análises estão sofrendo influência da aleatoriedade, que está relacionada ao acaso.

Para que você entenda esse efeito, existe um *site* feito para analista de dados rir. Chama-se *Spurious Correlations* e você pode acessar em https://www.tylervigen.com/spurious-correlations. Segundo o autor do *site*, existe uma correlação estatística entre o número de pessoas que se afogaram em piscinas com o número de aparições do ator Nicolas Cage em filmes ao longo dos anos. Será que os filmes do ator são tão ruins a ponto de influenciar pessoas a se afogarem em suas piscinas? Lógico que não, até porque eu gosto de muitos dos filmes dele e estou bem vivo. Um outro exemplo é a correlação entre o número de divórcios em Maine (Estados Unidos) e o consumo *per capita* de margarina. Qual é a relação causal desse absurdo? Pense em alguma saída e ria junto comigo.

Mas, o fato é que essas e outras relações encontradas e abordadas no *site* na verdade não passam do acaso. Quando trabalhamos com testes de hipótese, a princípio podemos achar uma relação entre as variáveis e o desfecho, contudo, não podemos afirmar categoricamente que uma determina ou causa a outra.

Se eu estudo um vírus novo na população humana e digo que ele é a causa de uma doença respiratória, eu preciso tomar muito cuidado nessa afirmação. O estudo é muito novo e talvez eu não tenha ainda a bagagem investigativa robusta para afirmar isso categoricamente. Segundo Fletcher (2008), quando os biomédicos estudam causa, normalmente eles buscam o mecanismo patogênico ainda ocultos à luz da ciência. Quando desvendado as vias finais comuns da doença, aí podemos afirmar sobre causalidade.

O efeito é justamente o resultado ou consequência de uma causa ou conjunto de causas. Refere-se ao impacto ou alteração que ocorre como resultado da presença ou influência das causas em um processo ou sistema. Dessa forma, causa e efeito têm grande proximidade, mas sua relação só pode ser afirmativa através de grandes evidências.

18.2 PREVALÊNCIA

A prevalência é um dos índices epidemiológicos mais utilizados. Prevalência é proporção do grupo de pessoas com uma determinada doença em uma população, num determinado período de tempo.

Exemplificando, podemos dizer que a prevalência de pessoas com hanseníase em um determinado município foi de 14 casos a cada 10.000 habitantes no ano de 2024. Temos a proporção: 14/10000; temos o tempo: 1 ano (ano de 2024). Então, chegamos à conclusão que a cada 10.000

moradores desse município, 14 apresentou a doença. Pela lógica, se o município tiver 20.000 habitantes, o total de doentes seria 28 nesse mesmo ano, concorda? Então, tudo é um jogo de proporcionalidade.

Podemos escrever uma expressão matemática simples para acharmos a prevalência:

$$Prevalência = \frac{número\ de\ casos\ no\ período}{total\ de\ habitantes\ no\ período}\ x\ fator\ multiplicador$$

Esse fator multiplicador ele é opcional. Alguns autores preferem chamar de constante. Dependerá do valor encontrado pela fração inicial. Vamos dar alguns exemplos e você perceberá quando ele é necessário. Vamos recalcular a prevalência acima sem o uso de um fator multiplicador. Temos um total de 20.000 habitantes e 28 apresentaram hanseníase no período de 1 ano. Vamos pôr na fórmula:

$$Prevalência = \frac{28}{20000} = 0{,}0014$$

O valor calculado significa que houve 0,0014 casos de hanseníase por habitante. O número do jeito que está me causa desconforto, pois não é um número fácil de interpretar. O que seria 0,0014 casos? Sendo um caso, uma pessoa com a doença, quanto é 0,0014 pessoa? Complicado, não?

Se multiplicarmos 0,0014 por 10.000, por exemplo, encontraremos o valor 14. Poxa, que alívio. Como o fator multiplicador foi 10 mil ou 10^4, podemos voltar a dizer que a prevalência foi de 14 casos a cada 10 mil habitantes nesse tempo de um ano. No R calcularíamos essa prevalência assim:

```
28/20000*10000
[1] 14
```

Se o fator multiplicador foi 100.000 (10^5), pela leia da proporção, a prevalência será 140 casos a cada 100.000 habitantes nesse período de 1 ano. Na realidade, esses valores são interpretativamente falando os mesmos de cima. Mas é preciso ter um equilíbrio, para o valor da prevalência não ficar elevado demais e ter interpretações espúrias. Por exemplo, para o fator multiplicador de 1 milhão (10^6) a prevalência seria 1.400 casos a cada 1.000.000 de habitantes no ano. Mas o município tem um milhão de habitantes? O primeiro valor (1.400) é o que chama mais atenção. Então, precisamos ficar atentos na informação que desejamos passar, e passar com imparcialidade.

Vamos para um segundo exemplo. Pretende-se calcular a prevalência de casos com HPV no Brasil, no mês de janeiro de 2024; supondo que o Ministério da Saúde deseje estudar a infecção para planejamento de projetos de intervenção em saúde da população, como vacinação e outras medidas preventivas. Segundo o IBGE, estima-se que a população brasileira seja de 127,3 milhões de pessoas com idade sexualmente ativa. Estudos prévios sugerem que haja 84 milhões de infectados pelo vírus no Brasil (são números fictícios, apesar de que as taxas de infeção por esse vírus serem realmente elevadas). Calculando a prevalência nacional de pessoas com idade sexualmente ativa, ainda sem o fator multiplicador, teremos:

$$Prevalência = \frac{84.000.000}{127.300.000} = 0{,}6598586$$

O resultado dessa divisão gera um número mais amigável. Se multiplicarmos por 100 (fator multiplicador), encontraremos 65,99%. Dessa forma, podemos dizer que a prevalência de infecção pelo HPV na população sexualmente ativa brasileira é de 65,99% ou 65,99 casos a cada 100 habitantes sexualmente ativos, no período de janeiro de 2024.

Como o período é relevante no cálculo da prevalência, há três classificações de prevalência quanto a isso. Essas classificações ajudam a compreender a evolução da prevalência das doenças nas populações em estudo. São elas:

- **Prevalência pontual**: é a proporção de casos existentes em um dado instante no tempo. Por exemplo, no primeiro dia de cada mês, no último dia de cada mês, ou ainda no dia 15 de cada mês.
- **Prevalência de período**: é a proporção de casos existentes em um período de tempo. Por exemplo, a prevalência durante um ano ou a prevalência durante um mês.
- **Prevalência na vida**: é proporção de pessoas que apresentaram pelo menos um episódio da doença em estudo ao longo da sua vida. Por exemplo, 1,03% das pessoas do município tal apresentaram a doença X ao longo de suas vidas.

Por fim, o curso temporal da doença, ou seja, a história natural da doença, só finalizará quando o paciente apresentar recuperação (cura) ou sua evolução seguir para o óbito. Enquanto isso, ele é um indivíduo prevalente da doença e contará nas taxas de prevalência.

Ao comparar qualquer indicador de saúde entre localidades diferentes, é obrigatório que o fator multiplicador seja exatamente o mesmo entre elas.

18.3 INCIDÊNCIA

A incidência é uma métrica que leva em consideração os casos novos. Por vezes alguns estudantes se confundem se uma determinada taxa é prevalência ou incidência. Para que não fique dúvida, há duas perguntas diferentes a serem respondidas: (1) qual a proporção de pessoas que tem uma determinada condição clínica? (2) Qual a taxa de novos casos num determinado tempo da referida condição?

No primeiro questionamento os gestores de saúde devem se preocupar mais com a manutenção da vida e cura das pessoas afetadas. É importante que você internalize que doença custa dinheiro (para o paciente e para os sistemas de saúde). Dessa forma, fazer com que as pessoas se curem ou estejam estabilizadas (nos casos de doenças que não tem cura) é a uma das maiores preocupações quando temos prevalências altas.

O segundo questionamento tem a ver com a prevenção. Quando temos altas taxas de casos novos na população, quer dizer que objetivos dos projetos de prevenção da doença foram "quebrados", ou por negligência, ou por uma nova cepa que surgiu, introdução de um microrganismo em uma população que não tinha contato histórico, enfim. Com isso, os casos novos de tempos em tempos começam a aumentar, indicando que os gestores em saúde precisão se preocupar em barrar as condições da gênese da doença, como a transmissão, por exemplo.

A Figura 18.1 demonstra em um esquema a dualidade que é a incidência e a prevalência. Nos estudos epidemiológicos elas "conversam" entre si. Imagine um tanque d'água onde a água são os indivíduos afetados. A torneira que enche o tanque são os casos novos e isso está vinculado à incidência. A água dentro do tanque, no período que estamos vendo, são os indivíduos prevalentes. Há duas torneiras que evadem a água do tanque, que indica o curso final da doença: cura ou óbito.

O cálculo da incidência pode ser feito usando a fórmula abaixo:

$$Incidência = \frac{número\ de\ casos\ novos\ no\ período}{total\ de\ expostos\ ao\ risco\ no\ período}\ x\ fator\ multiplicador$$

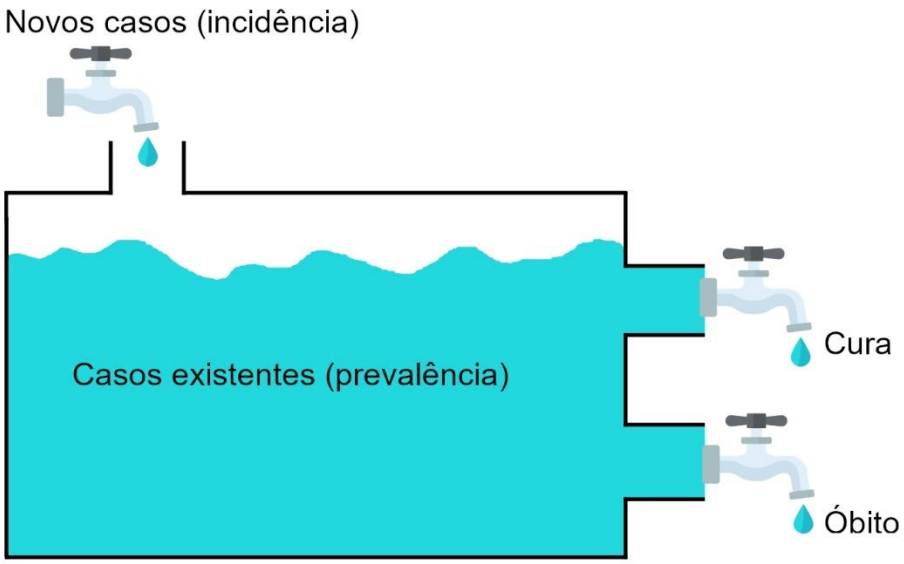


Figura 18.1. Esquema representativo da incidência e prevalência
Fonte: Demarchi e Chatalov, 2021 (modificado).

Perceba que o denominador aqui empregado é diferente do cálculo da prevalência. Na incidência estamos preocupados com a pessoas que têm o risco real de ficarem doentes por uma determinada condição. Por exemplo, se estamos calculando a taxa de incidência de alguma IST (infecção sexualmente transmissível) como HPV, HIV, sífilis, gonorreia e outras, vamos contabilizar as pessoas expostas ao risco, que são as pessoas de idade sexualmente ativas. Recém nascidos e idosos de idade muito avança não seriam o alvo desse tipo de estudo, por exemplo. Então, cada doença/condição terá sua população alvo no cálculo da incidência.

A nível demonstrativo, vamos supor que em um determinado município do interior brasileiro a quantidade de pessoas sexualmente ativas é 215.674 habitantes. Esses quantitativos são geralmente estabelecidos pelos censos. No mês de julho, pesquisadores percorreram os hospitais e clínicas pertencentes a esse município e encontraram 284 casos novos confirmados com sífilis. Vamos usar o fator multiplicador 10.000 por bom senso. Calculando a incidência de sífilis no hipotético município diretamente no R:

```
284/215674*10000
[1] 13.16802
```

Ou seja, a incidência é de aproximadamente 13 casos novos de sífilis a cada 10.000 habitantes em idade sexualmente ativa em junho.

Ainda hipoteticamente falando, vamos supor que as pessoas sexualmente ativas desse mesmo município em agosto é de 215.503 habitantes e que foi encontrado 305 casos novos. Recalculando a incidência do mês de agosto, teremos:

```
305/215503*10000
[1] 14.15294
```

Como resultado, temos que em agosto houveram aproximadamente 14 casos novos a cada 10.000 habitantes em faixa etária sexualmente ativa em agosto.

Aqui há uma instigação. Subir aproximadamente 1 caso novo de um mês para o outro será que é expressivo ou a doença está em estabilidade? Vou jogar-lhes um balde de água fria porque não vou lhe responder com precisão essa instigação. Porém, quero lhe dizer que há métodos analíticos bioestatísticos e epidemiológicos mais avançados que tratam do curso das doenças de forma temporalmente sequenciada. Esse tipo de entendimento foi jogado à tona no período da pandemia da COVID-19, onde as mídias populares falavam muito em média móvel. Na relação de métodos há o estudo de séries temporais, que daria um livro à parte. Se o seu estudo necessita de medições de prevalência e/ou incidência ao longo de vários períodos, pense em se aprofundar nesse tipo de método.

Para avaliar se uma doença está em uma crescente de casos, decrescente ou estabilidade, podemos utilizar a regressão linear proposto por Prais-Winsten. Um ótimo exemplo é o artigo de Silva e colaboradores (2022) titulado "Tendência temporal das taxas de incidência e de mortalidade por COVID-19 e sua relação com indicadores socioeconômicos no Piauí: estudo ecológico, 2020-2021". Você verá como as taxas de incidência são trabalhadas de tempos em tempos e há a avaliação de tendência.

18.4 SURTO, EPIDEMIA, PANDEMIA, ENDEMIA

Já que acabamos de falar em surgimento de casos novos, não podemos deixar de passar por quatro nomenclaturas importantíssimas na epidemiologia que caracterizam o curso de casos.

Primeiramente, vamos compreender o que é um surto. Segundo o dicionário Dicio online um surto é um impulso, uma arrancada. Na epidemiologia o sentido não é tão diferente assim. Um **surto** é a ocorrência repentina de casos de uma doença em um grupo de pessoas ou em uma área geográfica específica (geralmente um bairro de um município) que excede o que é normalmente esperado.

Os surtos podem ser causados por uma variedade de fatores, como a introdução de um novo agente patogênico em uma comunidade, a propagação rápida de uma doença infecciosa de pessoa

para pessoa em uma determinada localização, condições ambientais favoráveis para a disseminação do patógeno, ou a exposição a fontes comuns de contaminação.

Um exemplo de surto é a ocorrência do aumento significativo no número de casos de gripe em uma escola primária durante o inverno. Suponha que, em condições normais, apenas algumas crianças na escola adoecem com sintomas de gripe a cada semana. No entanto, durante uma semana específica, um grande número de alunos começa a apresentar febre, tosse, espirros e outros sintomas típicos. O número de crianças doentes supera claramente a média esperada. Neste cenário, as autoridades de saúde municipais podem considerar isso como um surto de gripe na referida escola ou no bairro a qual ela se localiza.

Quando falamos em **epidemia**, também estamos tratando dos aumentos repentinos de casos mais do que é esperado. Contudo, a localização geográfica não se restringe a apenas um único local. Se estamos falando de uma epidemia municipal, então está claro que alguns ou vários bairros apresentaram surtos coletivos. Quando falamos em epidemia de nível estadual é porque diversos estados da nação aumentaram seus números de casos novos rapidamente.

Como exemplo de epidemia, vamos supor que um determinado município costeiro tenha tido problemas gravíssimos com o despejo de esgoto no mar. Contando que todo o litoral desse município é explorado por pescadores e marisqueiras, diversos bairros foram afetados com aumento repentino de casos de hepatite A, supostamente através do consumo de frutos do mar contaminados pelo esgoto. Seria um caso de epidemia de nível municipal por hepatite A.

Quando uma epidemia se espalha por diversos locais do planeta, estamos diante de uma **pandemia**. Certamente é o pior dos cenários, uma vez que diversas populações entraram em contato do um agente patogênico e causou doença. Medidas mais rigorosas devem ser tomadas nesses casos, pois o curso da doença é mais incerto, uma vez que diversas etnias estarão sendo "testadas". É difícil saber porque algumas pessoas não sentem nenhum sintoma quando infectadas enquanto outras sucumbem ao óbito rapidamente. A ciência vem descobrindo diversos pontos sobre o assunto, inclusive com o uso da bioinformática, mas saber todas as interações moleculares entre hospedeiro e o microrganismo ainda é um desafio enorme.

Como exemplo, a gripe espanhola foi uma pandemia causada pelo vírus da influenza no ano de 1918 e 1919. A doença respiratória causou mais de 20 milhões de mortes pelo mundo e no Brasil se espalho quando ancorou nessas terras o navio inglês Demerara, que trouxe infectados da Europa. Dos portos de Salvador, Recife e Rio de Janeiro se espalho para regiões interioranas no país. Mais recentemente, também um vírus que gera problemas respiratórios gerou uma pandemia,

o SARS-CoV-2, que começou no ano de 2019 a se disseminar pelo mundo, causando a doença COVID-19. A infecção pelo HIV, por exemplo, também é considerada uma pandemia até os dias atuais.

Por fim, temos o termo chamado **endemia**, que não está diretamente relacionado à questão quantitativa de transmissão de mazelas. Quando uma doença é classificada como endêmica, é porque ela em algum momento já foi um surto ou epidemia e seus níveis se estabilizaram ao longo dos anos, mas não cessaram os casos, se tornando constante em uma localidade. Com isso, a doença se tornou típica dessa determinada região. No Brasil, um dos casos mais clássicos que podemos exemplificar são as endemias de febre amarela e malária na região Norte do país. A primeira é causada por um arbovírus, no qual o mosquito *Aedes aegypti* é o principal vetor de ciclo urbano, enquanto que os gêneros *Haemagogus* e o *Sabethes* são os principais vetores no ambiente silvestre. Já a malária é causada por protozoários parasitas do gênero *Plasmodium* e também tem como vetor mosquitos, porém do gênero *Anopheles*. Para quem não é da região Norte do Brasil e pretende visitá-la, é recomendado que se vacine antes da viagem para aproveitar as belezas únicas dessa região.

18.5 TIPOS DE ESTUDOS EPIDEMIOLÓGICOS

Na epidemiologia existem vários tipos de estudos que são usados para investigar diferentes aspectos de doenças. Cada tipo de estudo tem suas próprias vantagens e desvantagens, e a escolha depende da pergunta da pesquisa e dos recursos disponíveis.

Entre os diversos tipos, que abordaremos mais à frente, há uma ramificação primária de grande importância que são entre os grupos de estudos observacionais e os grupos de estudos experimentais:

- Estudos observacionais: são estudos em que os pesquisadores observam o efeito de um fator de risco, teste diagnóstico, tratamento ou outra intervenção sem tentar alterar quem está ou não exposto ao fator de risco. Eles são frequentemente usados para identificar razões que podem contribuir para uma condição médica ao comparar grupos de indivíduos que são ou não expostos ao fator em estudo. Eles podem ser descritivos ou analíticos.
- Estudos experimentais: são estudos em que os pesquisadores introduzem uma intervenção e estudam os seus efeitos. Estudos experimentais são, portanto, ensaios, seja de um medicamento novo, ou dosagem nova, de uma nova vacina, ou novos tratamentos de forma geral.

A Figura 18.2 resume os tipos de estudos epidemiológicos que estudaremos a seguir. Contudo, ainda na referida figura, de baixo para cima, temos uma certa gradação de níveis de evidência científica crescente. Isso indica que quanto mais próximo dos estudos descritivos, a verdade científica ainda pode ser muito discutida, testada e retestada. Quando chegamos próximo aos estudos de revisão sistemática, significa que o conhecimento trabalhado está o mais solidificado possível. Não que seja exatamente a verdade absoluta, mas que é o que melhor conseguimos produzir de conhecimento até o determinado momento.

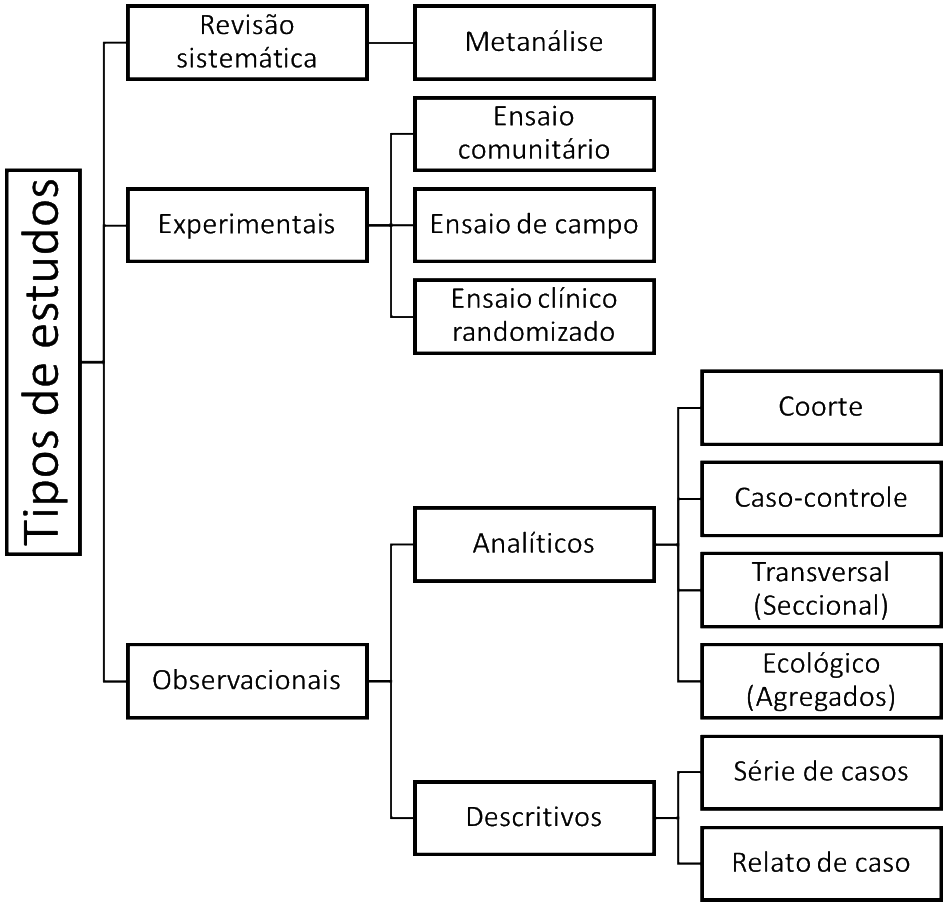


Figura 18.2. Tipos de estudos epidemiológicos.

18.5.1 Descritivos: Relato de caso e Série de casos

Comecemos a falar dos estudos observacionais descritivos. A essa classificação podemos ter um relato de caso, que é o tipo de estudo que se concentra em um indivíduo com uma condição clínica específica. Os relatos de caso são importantes para documentar doenças novas, não corriqueiras, manifestações raras, ou mesmo um curso raro da doença. Eles fornecem fontes para

futuras hipóteses sobre apresentação da doença, risco, prognóstico e tratamento alternativo. É o átomo da ponta do *iceberg* e por isso, é a menor unidade publicável na literatura em saúde ou área correlata.

A maior diferença entre relato de casos e uma série de casos é o fator quantitativo dos indivíduos em estudo. Geralmente os relatos de casos não passam de três indivíduos (alguns autores consideram até 9 indivíduos), enquanto que série de casos podemos descrever três ou mais indivíduos.

Como qualquer estudo, a escolha pelo relato ou série de casos haverá vantagens e desvantagens, as quais resumiremos:

- Vantagens: o detalhamento clínico, que oferece uma análise detalhada do diagnóstico clínico do indivíduo, incluindo características peculiares da normalidade, sintomas e sinais do paciente, como o paciente reagiu à terapia convencional e qual a terapia alternativa foi determinada;
- Desvantagens: limitação da amostra, pois compreende pouquíssimos casos, o que pode limitar a generalização dos resultados; e a falta de comparação, uma vez que se trata de um estudo descritivo, não realiza comparações diretas, mas sim o que ocorre na literatura, o que pode restringir a aplicabilidade dos achados em outras populações e também não podem comprovar a causa e o efeito.

Por exemplo, no início da pandemia do SARS-CoV-2, Rente e cols. (2020) publicaram um relato de caso titulado "Coronavírus e o coração: um relato de caso sobre a evolução da COVID-19 associado à evolução cardiológica", o qual descreve o caso de um paciente de 33 anos, com sinais e sintomas característicos de infecção viral do trato respiratório. É relatado os medicamentos administrados na primeira consulta, contudo o paciente evoluiu com piora alguns dias após. Ainda não se sabia a relação entre a infecção desse vírus com acometimentos de outros órgãos. Segundo os autores, o paciente apresentou taquicardia com abafamento de bulhas, dor torácica, turgência jugular com sinais de disfunção diastólica. Alguns marcadores bioquímicos da função cardíaca estavam alterados. Isso mostra o quanto os médicos sofreram nesse período para diagnosticar e tratar pacientes com COVID-19 grave, além ter sido a base para que estudos logo após esses relatos tratassem de investigar a relação entre COVID-19 e acometimentos cardíacos.

18.5.2 Estudo Analítico Ecológico

Um estudo ecológico é um tipo de estudo observacional onde a unidade de análise é a população ou mesmo ainda um grupo de indivíduos que pertencem a uma área geográfica (países, regiões, condados, municípios). Geralmente compara-se a ocorrência de uma doença ou condição de saúde e a exposição de interesse entre agregados de indivíduos (daí também serem chamados de estudo de agregados) para verificar a possível existência de associação entre elas.

Os estudos ecológicos avaliam como os contextos sociais e ambientais podem afetar a saúde de grupos populacionais. As medidas utilizadas no estudo ecológico geralmente são taxas ou proporções, indicadores sociais e epidemiológicos, incluindo características ambientais como aspectos climáticos, geológicos e geográficos.

Entre as vantagens e desvantagens desse tipo de estudo podemos relatar:

- Vantagens: são estudos rápidos e baratos, pois conseguimos dados da população de interesse geralmente pelas plataformas generalistas de indicadores de saúde e demográficos do governo; são bons para gerar novas hipóteses.
- Desvantagens: muito difícil de controlar fatores de confusão; medidas populacionais e não individuais, o que limita algumas análises estatísticas (a unidade de observação é o grupo e não o indivíduo); coleta de dados duvidosa, pois não dá para sabermos do rigor e padronização da coleta de dados.

Como as medidas são agrupadas e não temos os dados específicos de cada indivíduo, pode-se gerar um erro analítico que alguns chamam de “falácia ecológica”, que é justamente o equívoco de concluir a relação entre variáveis em nível individual quando na verdade é feita com base de dados agregados. Por exemplo, se avalio os estados de um país e observo que o consumo de café é maior onde as taxas de ansiedade também são elevadas, não posso concluir que café gera ansiedade nas pessoas. Se houve correlação entre as variáveis do meu estudo será um sinal de alerta para que estudos mais aprofundados sejam feitos no intuito de confirmar a hipótese levantada. Há correlação, mas ainda não há o efeito de causa.

Um exemplo de estudo observacional analítico ecológico foi o artigo gerado por Lima e colaboradores, titulado “Tendência temporal, distribuição e autocorrelação espacial da hanseníase no Brasil: estudo ecológico, 2011 a 2021”. Os autores coletaram dados do sistema de Informação de Agravos de Notificação (dados governamentais), os quais calcularam as taxas de detecção anual dos casos de hanseníase por 100 mil habitantes, no período de 10 anos consecutivos. Como achado

principal, detectaram que o Brasil tinha uma tendência crescente da incidência de casos, porém, houve decréscimo após o período de pandemia da COVID-19, tendo as regiões Norte, Centro-Oeste e Nordeste as maiores taxas da doença.

18.5.3 Estudo Analítico Transversal

Em epidemiologia, um estudo transversal é um tipo de pesquisa observacional que analisa a relação entre exposições (fatores de risco) e desfechos (como doenças) em um determinado ponto no tempo. Por isso, esses estudos são úteis para fornecer uma "fotografia" instantânea da situação de saúde de uma população em um momento específico. Inclusive por conta dessa característica de corte temporal, esses estudos podem também ser chamados de estudos seccionais, inclusive, são chamados em inglês de "cross-sectional study".

Por conta dessa fotografia da condição de saúde da determinada população, acabam sendo estudos importantíssimos nas avaliações de prevalência e por isso são capazes de responder: quantos pacientes estavam doentes no período estudado? E, perceba que não é preocupação desse tipo de estudo considerar as mudanças ao longo do tempo.

Uma grande diferença em relação aos estudos ecológicos é que nos estudos transversais a unidade de observação é o indivíduo e não grupos. Por conta disso, as relações aqui encontradas são mais significantes a uma possível causalidade ou fator de risco para a doença em análise, mas ainda não são adequados para estabelecer relações causais diretas devido à sua natureza de corte único no tempo.

Analiticamente falando, pode ser realizado estimação de médias, medianas ou proporções, testes de associação e correlação entre variáveis de interesse. Contudo, há vantagens e desvantagens analíticas:

- Vantagens: são simples e de baixo custo; são estudos de rápida coleta análise de dados; podem gerar uma gama de resultados; avaliação da prevalência de doenças e agravos; não precisa de acompanhamento dos indivíduos da pesquisa; podem identificar grupos de maior risco.
- Desvantagens: sem análise temporal, sendo assim não permite estabelecer relações de causa e efeito; não analisa a incidência da doença analisada; possíveis erros de classificação, uma vez que indivíduos doentes podem não ser mais doentes com o passar do

tempo, ou sua classificação de grau ou nível da doença pode mudar, ou mesmo indivíduos não doentes podem apresentar a doença no futuro.

Como exemplo, podemos comentar o artigo de Cazolari et al (2020), "Níveis de Burnout e Bem-Estar de Estudantes de Medicina: um Estudo Transversal". Nesse estudo os pesquisadores aplicaram dois questionários (MBI-SS e Whoqol-Bref) em alunos do primeiro ao sexto ano do curso de Medicina da Unifesp. O primeiro questionário avalia Burnout (distúrbio emocional caracterizado por sintomas de exaustão extrema) e o segundo para avaliar a qualidade de vida. Eles avaliaram normalidade de variáveis contínuas, estimaram médias e desvio padrão, analisaram correlações através do coeficiente de Pearson, compararam médias através da ANOVA e utilizaram correção de Bonferroni como teste *post hoc*. Como conclusão principal, eles verificaram uma prevalência de exaustão emocional de 5% entre os estudantes, apresentando os estudantes uma qualidade de vida boa, com níveis baixos a moderados de *Burnout*.

18.5.4 Estudo Analítico Caso-controle

Um estudo caso-controle é um tipo de pesquisa observacional onde os participantes são selecionados com base na presença (casos) ou ausência (controles) de uma determinada doença ou condição. Portanto, a unidade observacional são os indivíduos participantes. A partir daí, o estudo retrocede no tempo para investigar se os casos e controles foram expostos a potenciais fatores de risco ou causas da condição em questão. Então, é fundamentalmente um estudo retrospectivo e comparativo.

Suas comparações fundamentam-se na associação entre evento de interesse (desfecho) e preditores (variáveis independentes) e estabelecer os critérios e inclusão e exclusão rigorosamente é um passo essencial para esse tipo de estudo. Por exemplo, se uma pesquisa trata acerca da asma, é fundamental que tato casos e controles não tenham qualquer outro tipo de doença pulmonar, caso contrário estamos adicionando vieses de seleção à amostragem.

As principais funções dos estudos caso-controle é investigar fatores de risco ou fatores protetores, além de elucidar prognóstico de doenças. Sua métrica mais usual e importante é a razão de chance (*odds ratio*), que trabalharemos melhor mais à frente, além de comparações de médias e análises de associações entre variáveis categóricas, entre outras. Mas como qualquer estudo, há vantagens e desvantagens:

- Vantagens: geralmente são estudos baratos e rápidos, dependendo do desfecho e demais variáveis que entrarão em jogo; são úteis na investigação de doenças raras; podem ser analisados múltiplos fatores de risco.
- Desvantagens: suscetíveis a viés de seleção entre casos e controles; suscetíveis a viés de informação e fatores de confusão; não servem para determinar incidência; certas exposições podem ser de difícil avaliação e precisão de dados.

Exemplificando, Okuyama e colaboradores (2020) escreveram um estudo titulado "Intoxicações e fatores associados ao óbito por agrotóxicos: estudo caso controle, Brasil, 2017". Nesse texto os autores estavam preocupados mais com o prognóstico da intoxicação por agrotóxicos. Foram selecionados todos os pacientes com quadro de intoxicação que foram atendidos por CIATox e registrados no Sistema Brasileiro de Dados de Intoxicações (Datatox) no referido ano. Pacientes que evoluíram a óbito foram considerados casos, enquanto que os demais foram considerados controles. Como resultado, ser idoso (OR = 4,94; IC95%: 2,49-9,80), do sexo masculino (OR = 1,68; IC95%: 1,15-2,46), trabalhador agrário (OR = 2,20; IC95%: 1,15-4,24), ter tentado suicídio (OR = 13,27; IC95%: 6,48-27,19) e exposto a produtos extremamente tóxicos (OR = 2,77; IC95%: 1,84-4,16) apresentaram mais chances de óbito pós-intoxicação.

18.5.5 Estudo Analítico Coorte

Um estudo de coorte é do tipo observacional longitudinal, ou seja, estudo em que os mesmos indivíduos são observados repetidamente ao longo do tempo. O termo coorte vem do latim "cohortis" e significa parte de uma legião de soldados da unidade tática de infantaria do antigo Império Romano.

Nesse tipo de estudo, os indivíduos selecionados (coorte) são identificados e acompanhados no decorrer do tempo para determinar a ocorrência de eventos de interesse (expostos e não expostos), como o desenvolvimento de uma doença, a exposição a determinados fatores de risco, ou a resposta a uma intervenção. Por conta disso, é o tipo de estudo ideal para descrever a incidência de desfechos.

Os estudos de coorte são fundamentalmente prospectivos pelo fato da assistência temporal desses indivíduos. Contudo, dependendo do momento de exposição e início de acompanhamento, eles podem ser classificados em: (1) coorte prospectiva, onde os participantes são acompanhados a partir do momento da exposição; ou (2) coorte retrospectiva (ou coorte histórica), onde os dados

de exposição são coletados retrospectivamente e os participantes são acompanhados a partir desse ponto no tempo.

Devido à sua abrangência, esse tipo de estudo de coorte observacional envolve diversas variáveis. Podem ser realizadas análises de associação utilizando regressão ou qui-quadrado e teste Exato de Fisher, além de avaliar o grau de associação utilizando risco relativo (veremos à frente), comparação de médias entre expostos e não expostos, enfim.

Mas nem tudo é perfeito. Há vantagens e desvantagens:

- Vantagens: são úteis para doenças com longo período de latência (como silicose, câncer, infecção por HIV, entre outras); vários efeitos podem ser avaliados ao mesmo tempo; a incidência da doença pode ser mensurada; possibilidade de averiguar potenciais fatores de confusão.
- Desvantagens: pode ter alto custo; achados estatísticos podem demorar para ocorrer, pois são estudos longos; pela grande quantidade de informação possível pode gerar muitos dados faltantes; pode haver mudança na condição de exposição (não exposto pode virar exposto e vice-versa); critérios de diagnósticos de doenças podem ser alterados ao longo do tempo; perda de indivíduos durante o estudo.

Para exemplificar, Cailleaux-Cezar e colaboradores (2018) publicaram o artigo "Impacto do tabagismo na conversão de cultura e no desfecho do tratamento da tuberculose pulmonar no Brasil: estudo de coorte retrospectivo". Em seu trabalho, que é um estudo de coorte retrospectivo, eles pretendiam avaliar o impacto do tabagismo no tratamento da tuberculose pulmonar. Segundo seus achados principais, observaram que o tabagismo foi associado com a dificuldade de negativar a cultura microbiana depois de dois meses de tratamento (risco relativo = 3,58 [IC95%: 1,30-9,86]; p = 0,01), bem como ao desfecho de tratamento desfavorável (risco relativo = 6,29 [IC95%: 1,57-25,21]; p = 0,009). Além disso, eles relatam que houve associação entre tabagismo e cultura microbiana positiva ao final do 2° mês de tratamento entre os tabagistas atuais (p = 0.027).

18.5.6 Estudos Experimentais

Também chamados de intervencionais, os estudos experimentais são aqueles em que os pesquisadores intervêm ativamente em uma ou mais variáveis de interesse para avaliar seu efeito sobre um desfecho específico, como o desenvolvimento de uma doença ou avaliação do efeito de um novo medicamento. Esses estudos são projetados para investigar relações de causa e efeito, e

como há o acompanhamento dos indivíduos participantes, são estudos prospectivos. Por isso mesmo são mais robustos, pois permitem um controle mais rigoroso de variáveis confundidoras.

Os estudos experimentais geralmente envolvem a divisão dos participantes em dois grupos: o grupo intervenção (como o uso um medicamento, tratamento ou programa de prevenção, por exemplo) e o grupo controle, que não recebe a intervenção ou recebe um placebo. Os pesquisadores então comparam os desfechos nos dois grupos para determinar se a intervenção teve um efeito significativo.

Em se falando de testes de novos medicamentos, esses estudos não só determinam a eficácia dessa nova terapia, mas também avalia os efeitos colaterais e adversos. Por isso são importantes na fase de pré-comercialização.

Os estudos experimentais nada mais são do que ensaios e existem pelo menos três tipos mais recorrentes: ensaio clínico randomizado, ensaio comunitário e ensaio de campo.

No **ensaio clínico randomizado**, os participantes são aleatoriamente alocados em diferentes grupos de tratamento, sendo pelo menos um grupo que recebe o tratamento em estudo (grupo intervenção) e outro grupo que recebe um tratamento de controle, placebo ou padrão (grupo controle). Os procedimentos são geralmente realizados em locais controlados, como clínicas ou laboratórios na tentativa de controlar fatores ambientais externos.

A randomização é fundamental, pois ajuda a garantir que os grupos de tratamento sejam comparáveis em relação a características importantes, como idade, sexo, gravidade da doença, entre outras variáveis, reduzindo assim o viés de seleção e permitindo que quaisquer diferenças nos desfechos observados entre os grupos sejam atribuídas ao tratamento em estudo.

O **ensaio de campo** é um tipo de estudo experimental que ocorre em um ambiente natural ou no campo (*in loco*), fora de um ambiente controlado de laboratório. Esse tipo de estudo é comumente utilizado para avaliar o impacto de intervenções em condições do mundo real e geralmente são extensões ou sequenciamento de estudos de ensaio clínico randomizado.

Os ensaios de campo geralmente envolvem um grande número de participantes e podem ser mais representativos da realidade do que os ensaios realizados em laboratório. No entanto, eles podem ser mais difíceis de controlar e interpretar devido a variáveis externas não controladas que podem influenciar os resultados.

Já o **ensaio comunitário** é um tipo de estudo realizado em comunidades ou populações em vez de indivíduos específicos. Neste tipo de estudo, uma intervenção pode ser aplicada a um bairro,

vila, município, ou outra unidade populacional, em vez de apenas a indivíduos selecionados. Em muitos casos, pode ser uma extensão ou sequenciamento de um estudo de ensaio de campo.

Os ensaios comunitários são frequentemente usados em saúde pública para avaliar intervenções destinadas a melhorar a saúde da comunidade, como programas de educação em saúde, intervenções ambientais, campanhas de vacinação em massa, entre outros. Geralmente são mais complexos devido à natureza multifacetada das intervenções e às interações entre indivíduos na comunidade. No entanto, eles podem fornecer evidências mais valiosas sobre a eficácia e impacto de intervenções de saúde pública em larga escala.

De forma geral, os estudos experimentais têm as seguintes vantagens e desvantagens:

- Vantagens: coleta de informações mais detalhadas; doses rigorosamente administradas; placebo e cegamento podem reduzir distorções de resultados indesejados.
- Desvantagens: demorados; podem ter custos elevadíssimos; grandes amostragens; maior rigor possível com questões éticas.

Como exemplo de um estudo experimental, Bavaresco e Lucena (2022) avaliaram o efeito da terapia a laser de baixa potência no tratamento de úlcera venosas. Seu trabalho trata-se de um ensaio clínico randomizado, onde escolheram aleatoriamente pacientes com úlceras venosas cuja metade pertenceu ao grupo controle (tratamento tópico e compressivo da ferida) e grupo intervenção (terapia a laser de baixa potência adjuvante). Tal estudo avaliou 82 úlceras no período de 1 a 5 anos, e como resultado principal, encontraram melhora significativa tomando-se como base vários indicadores clínicos, concluindo que a terapia a laser melhora e reduz o tempo de regeneração tecidual.

18.5.7 Revisão Sistemática

A revisão sistemática é suprassumo do conhecimento científico e você vai entender o porquê. Não um tipo exclusivo da epidemiologia, mas de grande importância na área biológica.

Uma revisão sistemática é um tipo de estudo que busca reunir, avaliar e sintetizar todas as evidências disponíveis sobre uma questão específica de pesquisa de forma metodologicamente rigorosa e transparente. O objetivo principal é fornecer uma síntese imparcial e abrangente da literatura existente sobre um determinado tópico, seguindo um protocolo predefinido para minimizar viés e erros.

Para conduzir uma revisão sistemática, os pesquisadores seguem um processo rigoroso que inclui a formulação de uma pergunta biológica clara, a definição de critérios de inclusão e exclusão para a seleção dos estudos, a busca abrangente e sistemática da literatura, a avaliação crítica dos estudos incluídos e a síntese dos resultados.

As revisões sistemáticas são consideradas uma das formas mais confiáveis e robustas de evidência em pesquisa, especialmente quando seguidas por uma metanálise, que é a análise estatística combinada dos resultados dos estudos incluídos. Nesse ponto precisa-se de uma observação, pois alguns pesquisadores têm metanálise como sinônimo de revisão sistemática e isso não é verdadeiro, mas sim faz parte de.

Vamos verificar as vantagens e desvantagens desse tipo de estudo:

- Vantagens: nível de evidência científica mais elevada possível; base importante para tomada de decisões; baixo custo.
- Desvantagens: limitações de inclusão de estudos, uma vez que há heterogeneidade entre os estudos publicados na literatura; complexidade analítica, que também remete a heterogeneidade dos estudos trabalhados e ter que tirar uma conclusão dentro de uma gama de variações metodológicas.

Como exemplo, vamos discutir brevemente o trabalho publicado por Barufaldi e colaboradores, titulado “Risco de iniciação ao tabagismo com o uso de cigarros eletrônicos: revisão sistemática e meta-análise”. Os autores tinham como objetivo relacionar o uso de cigarros eletrônicos à inicialização ao tabagismo por meio de uma revisão sistemática com metanálise de estudos longitudinais. Eles utilizaram o MEDLINE, Embase, LILACS e PsycInfo como base de captura dos artigos científicos. Por fim, 25 trabalhos foram incluídos na pesquisa. A metanálise demonstrou que o uso de cigarro eletrônico aumentou em aproximadamente 3,5 vezes o risco de experimentação de cigarro convencional (RR = 3,42; IC95%: 2,81-4,15) e em mais de 4 vezes o risco de tabagismo atual (RR = 4,32; IC95%: 3,13-5,94). Eles concluem, portanto, que a liberação da comercialização dos dispositivos eletrônicos pode representar ameaça para as políticas públicas de saúde em nosso país.

18.6 MEDIDAS DE ASSOCIAÇÃO PARA ESTUDOS RETROSPECTIVOS

Estudos retrospectivos são investigações que buscam analisar eventos passados em relação à ocorrência de doenças, óbitos ou agravos em uma pulação. Eles olham para trás no tempo para

reunir dados e informações sobre exposições. Eles são úteis para avaliar a história natural de uma doença, entender a distribuição temporal de eventos e gerar hipóteses sobre as relações causais. Nesses estudos não há qualquer intervenção ou preocupação futura no desenrolar da doença.

Os estudos analíticos retrospectivos cuja unidade é um indivíduo, podem ser **transversais** ou **caso-controle**, e há duas medidas de associação possíveis: a razão de prevalência (RP) e a razão de chance (*odds ratio* - OR). É importante, antes de mais nada, informar o que retornará delas. Quando calculamos RP ou a OR, estamos verificando a probabilidade de um evento ocorrer sendo o indivíduo exposto ao possível fator de risco. Lembrando que isso ainda não é causalidade, mas a busca hipotética de elementos que podem deixar pessoas mais vulneráveis ou não a terem a doença ou condição.

18.6.1 Razão de chances (*odds ratio*)

Em capítulos anteriores, estudamos o uso da OR quando aprendemos sobre regressão logística e o grau de associação entre o desfecho e variáveis preditoras (independentes). Faremos aqui um estudo complementar, portanto. Podemos calcular a OR utilizado a tabela de diagnóstico 2x2 (Tabela 16.1) estudada em capítulos anteriores. Vamos remodela-la para chamarmos atenção a outras nomenclaturas que aqui serão dadas (mas a distribuição dos dados é exatamente a mesma).

Então, a Tabela 18.1 é a mesma Tabela 16.1, porém renomeada. Perceba que a condição desfavorável da variável coluna contínua à esquerda (sim). Já a condição desfavorável da variável linha está acima (sim).

Tabela 18.1. Tabela de contingência 2x2 renomeada.

		Desfecho (doença/condição)		
		Sim	Não	Total
Exposição	Sim	a	b	a + b
	Não	c	d	c + d
	Total	a + c	b + d	a + b + c + d

Para exemplificar nossos cálculos no R, vamos supor que um pesquisador esteja analisando fatores de risco para a ocorrência de pancreatite aguda. A variável dependente terá a informação do paciente ter tido pancreatite aguda ou não. Geralmente os pesquisadores trabalham com bancos de dados ricos em variáveis, as quais todas elas ou parte delas serão avaliadas se gerarão

probabilidades aumentadas ou reduzidas de ocorrer o desfecho. Em nosso exemplo, vamos avaliar se o consumo diário de bebida alcoólica traz uma chance aumentada ou reduzida de infarto agudo, ou mesmo não ter efeito. Para isso, vamos preencher a Tabela 18.1 com dados fictícios que resultarão na Tabela 18.2.

Tabela 18.2. Tabela de contingência para o uso de bebida alcoólica diariamente e pancreatite aguda.

		Pancreatite aguda		
		Sim	Não	Total
Álcool diário	Sim	584	561	1.145
	Não	281	1.094	1.375
	Total	865	1.655	2.520

Perceba que em nosso exemplo o pesquisador se ateve a dados coletados no passado. Nessa pesquisa não é preocupação dele acompanhar a evolução dos pacientes, pelo menos nessa fase. Talvez seja uma preocupação do grupo de pesquisa, mas perguntas biológicas referentes ao futuro não serão respondidas nesse trabalho inicial. Perguntas diferentes, trabalhos e análises diferentes.

Para calcularmos a OR devemos escrever a razão entre o produto dos dados concordantes e o produto dos dados discordantes presentes na tabela 2x2. Assim, chegas à fórmula:

$$Razão\ de\ chance = \frac{a\,.\,d}{b\,.\,c}$$

Calculando a OR em nosso exemplo, teremos:

$$Razão\ de\ chance = \frac{584\,.\,1094}{561\,.\,281} = 4{,}0529$$

No R podemos calcular "à mão" da seguinte forma:

```
(584*1094)/(281*561)
[1] 4.052854
```

Para interpretarmos os valores de OR, devemos tomar como base três marcos:

- OR = 1: indica que não há associação entre desfecho e o suposto fator de risco, uma vez que a chance de o desfecho ocorrer é a mesma entre expostos e não expostos;
- OR > 1: existe chance aumentada do desfecho ocorrer nos indivíduos expostos. Esse resultado expressa quantas vezes mais o risco de o desfecho ocorrer no grupo de expostos do que dos não expostos.

- OR < 1: indica que a chance do desfecho ocorrer entre os expostos é menor do que entre não expostos. Isso acaba sendo um fator protetor e não de risco.

Por conta disso, voltando ao nosso exemplo, podemos dizer que quem bebe álcool todos os dias, há uma chance de aproximadamente 4 vezes de apresentar pancreatite aguda em comparação a quem não bebe todos os dias. Na prática clínica, entenda que nem todos os alcoólatras apresentarão pancreatite aguda, porém, eles estão expostos fortemente a esse risco.

18.6.2 Razão de prevalência

A razão de prevalência (RP) também é uma medida de associação, porém, que deve ser utilizada em estudos transversais. Ela retorna, na realidade, o grau de associação de um determinado evento ocorrer na população, que nada mais é do que a divisão da prevalência entre os expostos e a prevalência entre os não expostos. Lembre-se que estudos transversais são ótimos para avaliar prevalência.

Assim, pode ser calculada segundo a fórmula abaixo baseada na Tabela 18.1:

$$Razão\ de\ Prevalência = \frac{prevalência\ entre\ expostos}{prevalência\ entre\ não\ expostos} = \frac{\frac{a}{a+b}}{\frac{c}{c+d}}$$

Interpretativamente falando é muito similar, porém não igual, à OR. Se:

- RP = 1: significa que a prevalência do evento de interesse é a mesma nos dois grupos que estão sendo avaliados. Em outras palavras, a exposição não tem efeito sobre o desfecho, pois a prevalência entre expostos é igual à prevalência entre não expostos.
- RP > 1: indica que a prevalência do evento de interesse é maior no grupo exposto em comparação com o grupo não exposto. Indicaria fator de risco.
- RP < 1: isso sugere que a prevalência do evento de interesse é menor no grupo exposto em comparação com o grupo não exposto. Indicaria um fator protetor.

Para exemplificar nossos cálculos no R, iremos supor que uma estudante de nutrição, orientada por uma nutricionista experiente, tinha em mãos dados alimentares de uma população de seu município. Para eles, padronizaram que consumir frutas regularmente deveria ser considerado o uso pelo menos quatro vezes na semana. Entre os participantes da pesquisa, alguns foram diagnosticados com câncer de estômago e outros não tinham esse desfecho e nem outros problemas estomacais. Após entrevista, chegou-se à Tabela 18.3 com resultados de frequência absoluta.

Tabela 18.3. Tabela de contingência dos casos de câncer de estômago e o consumo de frutas regularmente.

		Câncer estomacal		
		Sim	Não	Total
Consumo de frutas	Sim	117	537	654
	Não	354	142	496
	Total	471	679	1150

Calculando diretamente a RP no R:

```
(117/(117+537))/(354/(354+142))
[1] 0.2506609
```

Podemos dizer que o consumo regular de frutas é 0,25 vezes em quem apresenta câncer de estômago. Apesar de estar correta essa interpretação, não lhe causa um desconforto a afirmação? Podemos trabalhar esse resultado na forma percentual. Poderíamos dizer, a prevalência de câncer estomacal entre quem consome frutas regularmente é 25% da prevalência de quem não consome. Ou seja, não consumir frutas é mais prevalente entre quem apresentou câncer estomacal.

18.7 MEDIDAS DE ASSOCIAÇÃO PARA ESTUDOS PROSPECTIVOS

Um estudo epidemiológico prospectivo coleta dados sobre exposições e eventos de interesse à medida que o estudo ocorre, acompanhando uma população em um período futuro. Esses estudos são projetados para examinar a relação entre fatores de exposição e resultados de saúde, com o objetivo de identificar associações causais. Ao contrário dos estudos retrospectivos, que olham para eventos passados, os estudos prospectivos permitem uma avaliação mais direta e temporalmente sequenciada das relações entre as variáveis. Esses estudos geralmente envolvem a formação de uma coorte.

Além dos estudos de coorte, os ensaios clínicos também são essencialmente prospectivos, pois eles avaliam a eficácia de tratamentos que deverão ser assistidos ao longo de um tempo. Geralmente são conduzidos em fases distintas (fase I, II, III, e, em alguns casos, fase IV) para avaliar a eficácia do novo tratamento.

A medida de associação mais adequada aos estudos prospectivos é o risco relativo (RR). É através dessa métrica que inferimos causalidade, que, em outras palavras, indica o real fator de

risco a um determinado desfecho. Se você perceber, o risco relativo presente nos estudos prospectivos é certificador (ou por vezes refutador) dos estudos retrospectivos.

Podemos dizer que o RR é a relação entre a incidência no grupo exposto e a incidência no grupo não exposto. Matematicamente, podemos usar a seguinte expressão:

$$Risco\ Relativo = \frac{\frac{a}{a+b}}{\frac{c}{c+d}}$$

Se você perceber, a forma de calcular é a mesma da razão de prevalência, seguindo a distribuição dos dados em uma tabela 2x2. Mas o que muda? Todo o contexto. Os estudos são completamente diferentes entre si.

Vamos supor que um grupo de pesquisadores acompanham pacientes de um município, cuja preocupação maior era o surgimento de cardiopatias. Durante 30 anos os indivíduos foram avaliados quanto a diversos fatores: alimentares, comportamentais, uso de medicamentos, comorbidades, etc. Ao avaliar os casos de infarto agudo do miocárdio, contrapondo com quem não apresentou nenhum problema cardiológico, chegou-se à Tabela 18.4 a seguir quanto ao consumo de 50 gramas de açúcar diariamente.

Tabela 18.4. Tabela de contingência dos casos de infarto agudo do miocárdio e o consumo de 50 gramas de açúcar diariamente.

		IAM		
		Sim	Não	Total
Consumo de açúcar	> 50g	59	145	204
	≤ 50g	183	4259	4442
	Total	242	4404	4646

Calculando no R:

```
(59/(59+145))/(183/(183+4259))
[1] 7.020197
```

Dessa forma, concluiríamos que casos de infarto agudo do miocárdio são aproximadamente 7 vezes maiores quando as pessoas consomem uma dose de açúcar maior do que 50 gramas por dia em relação àquelas que consomem dosagens mais baixas ou igual a esse valor.

Perceba que a nível interpretativo a RR é muito similar à *odds ratio* e a razão de prevalência. Mais uma vez, o que muda é o tipo de estudo. É a forma e cuidado temporal da coleta dos

dados. Do trato com vieses. Da incerteza de escolher pacientes que não deveriam estar em determinados grupos no momento do estudo. Isso é fundamental nas frequências dos acometimentos e não acometimentos, o que influencia nos cálculos das métricas de associação.

Para fecharmos a questão interpretativa:

- RR = 1; não há associação entre a variável e o desfecho.
- RR > 1; indica risco; significa que o fator de exposição aumenta o risco de desenvolver o desfecho.
- RR < 1; indica proteção; significa que o fator de exposição diminui o risco de desenvolver o desfecho.

18.8 TRABALHANDO EPIDEMIOLOGIA NO R

Existem diversos pacotes do R que são voltados para solucionar problemas de cunho epidemiológico. É importante alertá-los que na epidemiologia existe uma bagagem de métodos matemáticos bastante robustos para descrever e analisar situações de saúde, mais complexas do que abordamos até agora. Então, não se assuste caso busque no CRAN do R por pacotes voltados para epidemiologia e você não entender muita coisa no que contiver nos manuais de tais pacotes. Certamente se tratará de métodos mais avançados, que se for a sua linha de estudo certamente se acostumará com isso com o passar do tempo. Contudo, aqui trabalharemos com o pacote chamado epiR, que apesar de ser robusto e complexo analiticamente, trabalharemos de forma simplificada com ele.

Assim como ocorre em outros pacotes da área, devemos antes de mais nada remodelar nossos dados originais em um vector numérico contendo os valores da tabela de diagnóstico (2x2). Esse passo é importante em diversas análises e devemos fazer separadamente das análises principais.

Vamos construir (ou importar) um banco de dados chamado epi2x2 o qual, de forma fictícia, traz informações de uma doença qualquer e uma variável de exposição também qualquer. Serão 150 observação, onde na variável de exposição o valor 0 indicará que a pessoa não está exposta e 1 quer dizer que a pessoa esteve exposta. Para a doença, 0 indicará a ausência dela e 1 a ocorrência dela.

```
set.seed(12345)
epi2x2 <- data.frame(
  exposicao = sample(c(0, 1), 150, replace = TRUE),
```

```
  doenca = sample(c(0, 1), 150, replace = TRUE)
)

#ou baixe o arquivo no Github e importe os dados

library(rio)
epi2x2 <- import('pasta_de_arquivo/epi2x2.txt')
```

Verificando como ficou nosso banco:

```
head(epi2x2)
  exposicao doenca
1         1      0
2         0      1
3         1      1
4         1      0
5         1      0
6         1      1
```

Podemos extrair a tabela de diagnóstico usando a função table(), mas tenha muito cuidado nesse momento, pois os resultados dessa função ainda não são exatamente a tabela de contingência da forma que trabalhamos ao longo do livro. Vejamos:

```
table(epi2x2)
         doenca
exposicao  0  1
        0 34 36
        1 42 38
```

Se você perceber, os valores estão invertidos, pois o R distribui os as classes na ordem alfanumérica. Vamos então recriar adequadamente a tabela de contingência:

Tabela 18.5. Tabela de contingência modifica a partir da função table().

	Doença	
Exposição	Sim (1)	Não (0)
Sim (1)	38	42
Não (0)	36	34

A partir disso, vamos criar um objeto que contenha esses valores, combinando-os na função c(), de tal maneira que o elemento "a" da tabela de contingência esteja em primeiro, o elemento "b" em segundo e assim por diante até o elemento "d", último. Ficará da seguinte forma:

```
dados_tab <- c(38,42,36,34)
```

Pronto, esse é formato do nosso "banco de dados" para analisarmos a *odds ratio* nos estudos caso-controle, razão de prevalência nos estudo transversais e risco relativo nos estudos de coorte, por exemplo.

Se você não tiver instalado o pacote epiR, instale-o e abra a biblioteca:

```
install.packages('epiR')
library(epiR)
```

Utilizaremos a função epi.2by2(), que é uma clara referência à tabela 2x2. Nesta função precisamos minimamente designar alguns parâmetros: "dat" indicaremos o nosso banco de dados modificado para tabela 2x2, ou seja, "dados_tab"; "method" indica o tipo de estudo que estamos trabalhando, que pode ser "case.control" para estudo de caso-controle, "cross.sectional" para estudos transversais, ou "cohort.count" para estudos de coorte; "interpret" é um parâmetro booleano (TRUE/FALSE), o qual podemos pedir para a função retornar a interpretação dos resultados ou não (vamos pôr TRUE para ver como ela interpretará os resultados obtidos).

Exemplo 1: caso-controle

Vamos supor que um oftalmologista esteja intrigado com o aumento de casos de miopia em seu consultório. Como ele faz parte de um grupo de pesquisa, o qual coletou dados de pessoas diagnosticadas com miopia e pessoas sem essa condição. Um fator de exclusão seria apresentar qualquer outro problema de visão. Ele deseja avaliar a hipótese de que o uso de colírios à base de corticoide esteja relacionado a quadros de miopia.

Na fase de análise dos dados, chegaram a uma tabela que resume a frequência absoluta entre essas duas variáveis. Ajustando esses dados como uma tabela de contingencia 2x2, temos a Tabela 18.6 a seguir:

Tabela 18.6. Tabela de contingência para casos de miopia em relação ao uso de colírios à base de corticoides.

Uso de colírios com corticoide	**Míope** Sim	Não
Sim	41	35
Não	27	39

Vamos transpor essas frequências para um vetor que se chamará "dados_tab1", seguindo rigorosamente a sequência das células da tabela de contingência 2x2: a, b, c, d.

```
dados_tab1 = c(41,35,27,39)
```

Pronto! Nosso "banco de dados" para a análises de associação e avaliar a intensidade dessa associação está pronto para ser usado na função epi.2by2(). Como o tipo de estudo é caso-controle, devemos pôr o argumento "case.control" para o parâmentro "method". Além disso, vamos solicitar para a função que o R retorne a interpretação dos resultados colocando como TRUE o parâmetro "interpret". Veja como fica:

```
epi.2by2(dat = dados_tab1,
         method = "case.control",
         interpret = TRUE)
```

Rodando esse comando, o R retornará os seguintes resultados:

```
          Outcome+  Outcome-   Total                  Odds
Exposed +       41        35      76    1.17 (0.73 to 1.81)
Exposed -       27        39      66    0.69 (0.40 to 1.13)
Total           68        74     142    0.92 (0.65 to 1.29)

Point estimates and 95% CIs:
-------------------------------------------------------------------
Exposure odds ratio                          1.69 (0.87, 3.30)
Attrib fraction (est) in the exposed (%)    40.68 (-21.24, 71.27)
Attrib fraction (est) in the population (%)  24.66 (-8.40, 47.64)
-------------------------------------------------------------------
Uncorrected chi2 test that OR = 1: chi2(1) = 2.406 Pr>chi2 = 0.121
Fisher exact test that OR = 1: Pr>chi2 = 0.133
 Wald confidence limits
 CI: confidence interval
 Measures of association strength:
 The exposure odds among cases was 1.69 (95% CI 0.87 to 3.3) times greater than expo-
sure odds among controls.

 Measures of effect in the exposed:
 40.7% of outcomes in the exposed were attributable to exposure (95% CI -21.2% to
71.3%).

 Measures of effect in the population:
 24.7% of outcomes in the population were attributable to exposure (95% CI -8.4% to
47.6%).
```

A função epi.2by2() retorna muita informação. Vamos filtrar esses resultados para o que nos interessa. Primeiramente, devemos verificar se a nossa entrada de dados foi correta. Perceba que nos resultados o R retorna a tabela 2x2. Outcome + é o resultado positivo da pesquisa, que no exemplo é apresentar miopia (casos). Outcome - significa os indivíduos que não apresentam miopia (controles). Exposed + indica os indivíduos que estão expostos ao colírio, pois fazem uso dele. Exposed - são os indivíduos que não usam os colírios à base de corticoides. Verifique que essa tabela está exatamente distribuída como a Tabela 18.6 acima.

Para avaliarmos se existe associação entre as variáveis, vamos buscar os valores de P. Nos resultados há dois: o teste de qui-quadrado de independência (em *Uncorrected chi2 test*) e o teste exato de Fisher (em *Fisher exact test*). O p-valor do qui-quadro foi 0,121 e o p-valor para o teste exato de Fisher foi 0,133. Com não são significativos, podemos dizer que não houve evidências suficientes para afirmarmos que o uso de colírios a base de corticoides é um fator de risco para a miopia.

A *odds ratio* está presente em *Exposure odds ratio* e seu valor foi de 1,69, com intervalo de confiança (IC) entre 0,87 e 3,30. Como esse intervalo está contido o valor 1, não podemos mensurar de forma alguma o grau de associação (que acabamos de verificar que não existe em nosso exemplo).

Se não há associação entre as variáveis analisadas, devemos desconsiderar a parte interpretativa dos resultados, que está em "Measures of association strength" em diante. Nos exemplos abaixo vamos avaliar essa parte dos resultados, não se preocupe.

Lembrete: análises de estudos caso-controle podem ser feitos através de uma regressão logística, cujos resultados trarão o p-valor da associação, a *odds ratio* e seu intervalo de confiança.

Exemplo 2: estudo transversal

Vamos supor que um dentista esteja investigando a prevalência de problemas de saúde bucal em crianças. Ele e sua equipe de pesquisa realizam exames odontológicos em crianças de fase escolar de seu município. Ele avalia problemas de cárie, gengivite, coleta dados de hábitos alimentas (quais alimentos a criança costuma comer), hábitos comportamentais (horários de escovação dentária, quantas vezes faz isso por dia, qual escova usa, qual pasta dental usa, entre outros).

Trata-se de um estudo transversal, uma vez que o odontologista está preocupado nas prevalências das ocorrências. Logicamente ele não irá avaliar apenas as frequências absolutas e relativas dos acometimentos e hábitos tabulados, mas avaliar possíveis fatores de risco tomando-se como base as prevalências das demais variáveis em frente aos problemas dentários.

Vamos imaginar que ele deseje avaliar a associação entre duas variáveis presentes no seu banco de dados: problemas dentários (dicotomizada: sim/não) e ingestão diária de doces e achocolatados (dicotomizada: sim/não). Chegaram nos resultados parciais da Tabela 18.7 a seguir:

Tabela 18.7. Tabela de contingência para problemas dentários e ingestão diária de doces e achocolatados.

Ingestão de doces e achocolatados	**Problemas dentários**	
	Sim	Não
Sim	14	8
Não	19	82

Vamos passar essas informações para um vetor que chamaremos de "dados_tab2". Ele conterá os valores da Tabela 18.7, dessa forma:

```
dados_tab2 <- c(14,8,19,82)
```

A partir desse vetor, vamos avaliar se essas variáveis estão associadas e qual o grau de associação. Lembrando que para estudos transversais precisamos indicar "cross.sectional" para o parâmetro "method". Veja:

```
epi.2by2(dat = dados_tab2,
         method = "cross.sectional",
         interpret = TRUE)


           Outcome+  Outcome-  Total              Prev risk *
Exposed +        14         8     22   63.64 (40.66 to 82.80)
Exposed -        19        82    101   18.81 (11.72 to 27.81)
Total            33        90    123   26.83 (19.24 to 35.57)

Point estimates and 95% CIs:
-------------------------------------------------------------------
Prev risk ratio                              3.38 (2.02, 5.65)
Prev odds ratio                              7.55 (2.77, 20.57)
Attrib prev in the exposed *                44.82 (23.33, 66.32)
Attrib fraction in the exposed (%)          70.44 (50.59, 82.31)
Attrib prev in the population *              8.02 (-2.91, 18.94)
Attrib fraction in the population (%)       29.88 (10.37, 45.15)
-------------------------------------------------------------------
Uncorrected chi2 test that OR = 1: chi2(1) = 18.489 Pr>chi2 = <0.001
Fisher exact test that OR = 1: Pr>chi2 = <0.001
 Wald confidence limits
 CI: confidence interval
 * Outcomes per 100 population units

 Measures of association strength:
 The outcome prevalence risk among the exposed was 3.38 (95% CI 2.02 to 5.65) times
greater than the outcome prevalence risk among the unexposed.

 The outcome prevalence odds among the exposed was 7.55 (95% CI 2.77 to 20.57) times
greater than the outcome prevalence odds among the unexposed.
```

```
 Measures of effect in the exposed:
 Exposure changed the outcome prevalence risk in the exposed by 151.83 (95% CI 59.57
to 244.09) per 100 population units. 86.8% of outcomes in the exposed were attributa-
ble to exposure (95% CI 71.5% to 93.7%).

 Number needed to treat for benefit (NNTB) and harm (NNTH):
 The number needed to treat for one subject to benefit (NNTB) is 2 (95% CI 2 to 4).

 Measures of effect in the population:
 Exposure changed the outcome prevalence risk in the population by 13.5 (95% CI -2.78
to 29.78) per 100 population units. 36.8% of outcomes in the population were at-
tributable to exposure (95% CI 29.7% to 44.7%).
```

Primeiramente, vamos verificar se a entrada de dados está correta verificando a tabela 2x2 que o R retornou. E sim, está tudo em ordem.

Sobre a associação entre a variáveis, tanto para o qui-quadrado (em *Uncorrected chi2 test*) quanto para o teste exato de Fisher (em *Fisher exact test*) os p-valores são < 0,001, o que indica uma forte associação. Lembrando que para inferirmos associação nesses testes, o p-valor precisa estar abaixo ou igual a 0,05 quando consideramos o nível de significância de 5%.

Devemos nos lembrar também que para estudos transversais é mais adequado avaliarmos a razão de prevalência (RP), que em nossos resultados está descrita em "Prev risk ratio". Dessa forma, a RP foi de 3,38, com intervalo de confiança entre 2,02 e 5,65.

Uma observação: perceba que se utilizássemos equivocadamente a *odds ratio* (presente em "Prev odds ratio" = 7,55 e IC entre 2,77 e 20,57) para estimar o grau de associação entre as variáveis estaríamos superestimando essa medida.

Na parte interpretativa dos resultados, especificamente em *Measures of association strength*, ele indica que o risco de prevalência do desfecho entre os expostos foi 3,38 (IC 95%: 2,02-5,65) vezes maior do que o risco de prevalência do desfecho entre os não expostos. Contextualizando, podemos dizer que o risco de apresentar problemas dentários em criança que consomem doces e achocolatado é 3 vezes maior do que aquelas crianças que não consomem esses produtos.

Exemplo 3: coorte

Como conversamos mais acima, os estudos prospectivos avaliam o risco relativo (RR), cujo cálculo é similar ao da razão de prevalência. O que muda entre um estudo e outro é a forma como os estudos são conduzidos. Nos estudos prospectivos os participantes da pesquisa são acompanhados durante um bom tempo. Isso dá para avaliar com maior segurança a relação entre o desfecho e outras variáveis. Já os estudos transversais coletam dados em um momento do tempo

passado, sem se preocupar se a condição atual do participante. Veja que nesse ponto acaba surgindo possíveis vieses de classificação para os estudos transversais.

Pois bem, vamos supor que uma equipe de endocrinologistas acompanhem pacientes de diversos pontos do país, trabalhando assim com um estudo de coorte (prospectivo). Uma das avaliações que eles desejam verificar é se a prática de exercícios físicos é uma exposição de risco, de proteção ou não influencia no acometimento de diabetes mellitus do tipo II.

Após a coleta de dados e primeiras análises descritivas, chegaram à Tabela 18.8 abaixo:

Tabela 18.8. Tabela de contingência para a relação entre diabetes mellitus do tipo II e práticas regulares de exercício físico.

Prática de exercícios físicos	**Diabetes do tipo II**	
	Sim	Não
Sim	571	3795
Não	3258	1179

Vamos transpor a Tabela 18.8 para um vetor que chamaremos de "dados_tab3". Como o estudo acima é uma coorte, no parâmetro "method" da função epi.2by2() devemos colocar o argumento "cohort.count". Vamos rodar os comandos.

```
dados_tab3 <- c(571,3795,3258,1179)

epi.2by2(dat = dados_tab3,
         method = "cohort.count",
         interpret = TRUE)


           Outcome+  Outcome-   Total              Inc risk *
Exposed +       571      3795    4366   13.08 (12.09 to 14.11)
Exposed -      3258      1179    4437   73.43 (72.10 to 74.72)
Total          3829      4974    8803   43.50 (42.46 to 44.54)

Point estimates and 95% CIs:
-------------------------------------------------------------------
Inc risk ratio                                0.18 (0.16, 0.19)
Inc odds ratio                                0.05 (0.05, 0.06)
Attrib risk in the exposed *                -60.35 (-61.99, -58.71)
Attrib fraction in the exposed (%)         -461.45 (-507.29, -419.06)
Attrib risk in the population *             -29.93 (-31.59, -28.27)
Attrib fraction in the population (%)       -68.81 (-71.99, -65.70)
-------------------------------------------------------------------
Uncorrected chi2 test that OR = 1: chi2(1) = 3261.087 Pr>chi2 = <0.001
Fisher exact test that OR = 1: Pr>chi2 = <0.001
 Wald confidence limits
```

```
 CI: confidence interval
 * Outcomes per 100 population units

 Measures of association strength:
 The outcome incidence risk among the exposed was 0.18 (95% CI 0.16 to 0.19) times
less than the outcome incidence risk among the unexposed.

 The outcome incidence odds among the exposed was 0.05 (95% CI 0.05 to 0.06) times
less than the outcome incidence odds among the unexposed.

 Measures of effect in the exposed:
 Exposure changed the outcome incidence risk in the exposed by -60.35 (95% CI -61.99
to -58.71) per 100 population units. -461.4% of outcomes in the exposed were at-
tributable to exposure (95% CI -507.3% to -419.1%).

 Number needed to treat for benefit (NNTB) and harm (NNTH):
 The number needed to treat for one subject to be harmed (NNTH) is 2 (95% CI 2 to 2).

 Measures of effect in the population:
 Exposure changed the outcome incidence risk in the population by -29.93 (95% CI -
31.59 to -28.27) per 100 population units. -68.8% of outcomes in the population were
attributable to exposure (95% CI -72% to -65.7%).
```

Interpretando os resultados, primeiramente verificamos que houve associação entre ser diabético e a questão do exercício físico (qui-quadrado: P< 0,001; teste exato de Fisher: P< 0,001).

Já que há associação, vamos verificar a força desta, avaliando o risco relativo presente em "Inc risk ratio" (RR = 0,18, IC95%: 0,16-0,19). "Inc" se refere a incidência; taxa de risco de incidência. Percebemos que o valor 1 não está contido no IC e o valor de RR é menor do que 1. Isso se configura fator de proteção, portanto.

Para a parte interpretativa desse valor, a função retornou para a medida da força de associação, que a incidência do desfecho entre os expostos foi 0,18 vezes a da incidência do desfecho entre os não expostos. Em outras palavras, o risco de ter diabetes do tipo II entre quem faz exercício físico foi 0,18 vezes do que entre quem não faz exercício físico. Ou, o risco de ter diabetes do tipo II entre quem faz exercício físico foi 18% menor em relação a quem não faz exercício físico. Ou, o risco de ter diabetes do tipo II entre quem não faz exercício físico é 72% maior em relação a quem faz exercício físico.

Capítulo 5

Apresentação de Resultados

Agora que passamos pelo conhecimento de diversas análises estatísticas e epidemiológicas, chegou a hora de conversarmos como apresentar da melhor forma os nossos resultados obtidos.

Ao longo da carreira, ouvi meus orientadores dizendo: “venda seu peixe”. Eles estavam certos. Ciência é mais do que resultados em uma bancada, é *marketing* também. Bons resultados engavetados ou mal apresentados passam desapercebidos e desvalorizados pelos leitores ou interlocutores. Por isso, é preciso ter em mente que a apresentação de resultados é fase importante da divulgação do conhecimento.

Ao se trabalhar com artigos científicos, trabalhos de conclusão de curso ou relatórios, é imteressante que os seus resultados estejam objetivos e de fácil compreensão para quem irá lê-los e interpretá-los. Tabelas e gráficos são as principais maneiras de apresentarmos os principais resultados. Fora eles, há quadros (que parecem tabelas fechadas), fotos, vídeos, imagens de exames como uma tomografia computadorizada ou raio X, áudio, enfim. Existem mídia diversas, mas repito, tabelas e gráficos são as principais e vamos abordar a partir de agora as diversas possibilidades de cada um.

19.1 TABELAS

Uma tabela é uma estrutura de dados organizada em linhas e colunas, formando uma grade onde as informações são dispostas de maneira tabular. Cada célula da tabela contém um valor único, na forma de palavras e números. As tabelas são comumente utilizadas para organizar e representar dados de forma sistemática, facilitando a leitura e sua interpretação.

Cada tabela deve ter seu título (informando do que se trata as informações contidas na tabela), seu corpo (as informações em si, que estão entre linhas e colunas), o cabeçalho (identificando o título de cada coluna), e, quando necessário, fonte (citação de onde vieram os dados) e/ou nota (alguma informação relevante, como abreviações ou tipo de teste estatístico realizado, por exemplo).

Outra característica interessante das tabelas é que elas não são “fechadas”. As linhas de bordas laterais não devem aparecer. Caso uma representação de dados onde haja linhas superior, inferior e laterais, isso nós chamamos de quadro (Quadro 19.1). Compare com a Tabela 19.1.

Tabela 19.1. Tamanho médio de pápula para os principais alérgenos em uma população X.

Alérgeno testado	N	Tamanho médio de pápula (mm)
Ácaro	138	6,1
Fungo	134	4,2
Gramínea	154	1,5
Mariscos	135	3,8

Fonte: Criado pelo autor.

Quadro 19.1. Tamanho médio de pápula para os principais alérgenos em uma população X.

Alérgeno testado	N	Tamanho médio de pápula (mm)
Ácaro	138	6,1
Fungo	134	4,2
Gramínea	154	1,5
Mariscos	135	3,8

Fonte: Criado pelo autor.

No R existe duas funções básicas para trabalhar com tabelas: table() e prop.table(). Já utilizamos elas ao longo do livro, mas vamos enfatizar algumas coisas aqui. Com table() nós verificamos as frequência absolutas, enquanto que prop.table() retornará as frequências relativas. Essas duas funções são extremamente úteis quando precisamos utilizar seus resultados em outras funções, em outras análises, ou meramente verificar como estão as coisas de forma rápida. Porém, como representação de resultados em um relatório ou trabalho científico, elas são horríveis e grosseiras demais para isso.

Primeiramente, vamos criar dois vetores simulando duas variáveis categóricas: a e b. Vamos solicitar ao R 50 observações que contenham valores apenas 0 ou 1 (simulando resultados dicotomizados: 0 = ausente; 1 = presente).

```
set.seed(321)
a <- sample(0:1, 50, replace = TRUE)
b <- sample(0:1, 50, replace = TRUE)

a
 [1] 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 0 0 1 0 0 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1
0 1 0 0 0 0 1 0 0 1

b
 [1] 0 1 1 0 0 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1
1 1 1 1 0 1 0 0 0 1

table(a, b)
   b
a    0  1
  0 12 11
  1 15 12
```

Perceba que ao usar a função table(), o primeiro parâmetro foi identificado como a variável linha e o segundo parâmetro foi a variável coluna. Observe que 12, 11, 15 e 12 são os valores das frequências absolutas. Vamos usar agora a função prop.table(), que deverá ter a função table() como argumento.

```
prop.table(table(a, b))
   b
a      0    1
  0 0.24 0.22
  1 0.30 0.24
```

Veja que agora o R retornou as frequências relativas na forma decimal. Na forma que foi descrita a função prop.table() ela retornará sempre a fração específica de cada célula da tabela. Verifique que a suma de todas as células dará o valor 1. Posso representar esses valores na forma de porcentagem, multiplicando todo o código acima por 100.

```
prop.table(table(a, b)) * 100
   b
a    0  1
  0 24 22
  1 30 24
```

Agora nós temos 24%, 22%, 30% e 24%. Somando tais valores, encontraremos 100%. Entretanto, na bioestatística essa forma de apresentação é mais rara de se fazer. Geralmente nós calculamos a porcentagem para que o 100% seja considerado nas linhas ou colunas. Para fazer isso, devemos especificar o parâmetro "margin" com o argumento 1 (para linhas) ou 2 (para colunas). Perceba a diferença.

```
#considerando as linhas
prop.table(table(a, b), margin = 1) * 100
   b
a          0        1
  0 52.17391 47.82609
  1 55.55556 44.44444
```

Perceba que a soma das linhas será 100%.

```
#considerando as colunas
prop.table(table(a, b), margin = 2) * 100
   b
a          0        1
  0 44.44444 47.82609
  1 55.55556 52.17391
```

Perceba que agora a soma das colunas é que será 100%.

Qual utilizar? Linhas ou colunas? Bem, depende de você, de como você deseja interpretar os resultados. Contudo, geralmente os pesquisadores preferem a forma de linhas, mas isso não é uma regra.

19.1.1 Tabelas elegantes

Existem diversos pacotes para R cuja finalidade é retornar tabelas melhores acabadas do que as funções acima. Usaremos um que considero ser atualmente o melhor e mais bem elaborado pacote para retornar tabelas já prontas para você exportar para o seu artigo científico ou trabalho acadêmico, é o gtsummary. Vamos instalá-lo e abrir a sua biblioteca:

```
install.packages('gtsummary')
library(gtsummary)
```

Esse pacote contém um banco de dados chamado "trial", que é um conjunto de dados contendo as características de 200 pacientes que receberam o medicamento A (*Drug A*) ou o medicamento B (*Drug B*) presente na variável "trt" (*Chemotherapy Treatment*). O conjunto de dados também contém idade (variável age), um marcador sanguíneo (marker), estadiamento do tumor (stage), classificação histológica do câncer (grade), resposta ao tratamento (response), se o paciente evoluiu a óbito (death) e a quantidade de meses até o óbito (ttdeath). Vamos carregar o banco e verificar suas variáveis:

```
data(trial)

trial
# A tibble: 200 × 8
   trt      age marker stage grade response death ttdeath
   <chr>  <dbl>  <dbl> <fct> <fct>    <int> <int>   <dbl>
 1 Drug A    23  0.16  T1    II           0     0    24
 2 Drug B     9  1.11  T2    I            1     0    24
 3 Drug A    31  0.277 T1    II           0     0    24
 4 Drug A    NA  2.07  T3    III          1     1    17.6
 5 Drug A    51  2.77  T4    III          1     1    16.4
 6 Drug B    39  0.613 T4    I            0     1    15.6
 7 Drug A    37  0.354 T1    II           0     0    24
 8 Drug A    32  1.74  T1    I            0     1    18.4
 9 Drug A    31  0.144 T1    II           0     0    24
10 Drug B    34  0.205 T3    I            0     1    10.5
# ℹ 190 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Vemos que ele se apresenta como um *tibble*, que é uma espécie de *data frame*, que apresenta 200 observações e 8 variáveis. Além disso, há variáveis numéricas do tipo contínua (dbl) e inteiro (int), categóricas do tipo caractere (chr) e do tipo fator (fct). Algumas variáveis apresentam dados faltantes (NA).

A função que iremos utilizar para construir nossas tabelas será a tbl_summary(). Vamos trabalhar com as três primeiras variáveis do banco de dados e a variável que indica o óbito (sétima), criaremos um novo objeto após a seleção de variáveis e colocaremos ele como argumento da

função tbl_summary(). Por fim, vamos verificar como sairá a tabela com os resultados sumarizados com essas variáveis (Tabela 19.2, que aparecerá na aba *Viewer* do RStudio).

```
#Selecionando variáveis
trial2 <- trial[c(1:3,7)]

#Tabela 19.2
tbl_summary(trial2)
```

Tabela 19.2. Resumo dos resultados de cada variável presente em trial.

Characteristic	**N = 200**[1]
Chemotherapy Treatment	
Drug A	98 (49%)
Drug B	102 (51%)
Age	47 (38, 57)
Unknown	11
Marker Level (ng/mL)	0.64 (0.22, 1.39)
Unknown	10
Patient Died	112 (56%)

[1]n (%); Median (IQR)

Como resultado, a Tabela 19.2 já é muito interessante, indiscutivelmente melhor do que é gerado por table() e prop.table(). Ela apresenta notas, demonstra o título das colunas em destaque, quantifica o total de indivíduos, as variáveis categóricas apresentam frequência absoluta e relativa (%) e as variáveis numéricas foram descritas com a mediana e o intervalo interquartil.

Contudo, dá para melhorar bastante e mudar alguns padrões. Primeiramente, todas as variáveis estão em inglês devido à origem do banco de dados. Podemos a qualquer momento modificar o nome das variáveis na tabela usando o parâmetro “label” e colocar os nomes da forma desejada em uma lista, da maneira como vemos na Tabela 19.3.

```
#Tabela 19.3
tbl_summary(trial2,
            label = list(trt ~ "Tratamento",
                         age ~ "Idade",
                         marker ~ "Marcador sérico",
                         death ~ "Óbitos"))
```

Tabela 19.3. Modificação dos nomes das variáveis.

Characteristic	**N = 200**[1]
Tratamento	
Drug A	98 (49%)
Drug B	102 (51%)
Idade	47 (38, 57)
Unknown	11
Marcador sérico	0.64 (0.22, 1.39)
Unknown	10
Óbitos	112 (56%)

[1]n (%); Median (IQR)

Sobre as classes em cada variável categórica, perceba que para "trt", que estavam descritas no banco de dados como "Drug A" e "Drug B", apareceram as duas classes na tabela. Já para a variável "death", que apesar de ser categórica, mas está na forma *dummy* (dicotomizada) com os valores 0 e 1, a tabela apresenta apenas as frequências do maior valor (1), pois a função entende que esse é a característica de interesse.

Colocar na tabela apenas a classe de interesse pode ser interessante para economizar espaço quando temos muitas linhas, mas não pode ficar dúvidas sobre a classe que está omitida.

Para modificar o nome das classes das variáveis categóricas, podemos modifica-las no próprio banco de dados usando a função ifelse(), por exemplo. Observe o código e verifique como muda a nossa tabela original (Tabela 19.4).

```
trial2$trt <- ifelse(trial2$trt == 'Drug A', 'Droga A', 'Droga B')
trial2$death <- ifelse(trial2$death == 1, 'Óbitos', 'Vivos')

#Tabela 19.4
tbl_summary(trial2,
            label = list(trt ~ "Tratamento",
                         age ~ "Idade",
                         marker ~ "Marcador sérico",
                         death ~ "Óbitos"))
```

Tabela 19.4. Modificação dos nomes das classes das variáveis categóricas.

Characteristic	**N = 200**[1]
Tratamento	
Droga A	98 (49%)
Droga B	102 (51%)
Idade	47 (38, 57)
Unknown	11
Marcador sérico	0.64 (0.22, 1.39)
Unknown	10
Óbitos	
Óbitos	112 (56%)
Vivos	88 (44%)

[1]n (%); Median (IQR)

As tabelas acima são comumente utilizadas para descrever a amostragem. Geralmente, ela é uma das primeiras partes dos resultados. Contudo, quando estamos analisando relações entre desfecho e outras variáveis, a variável dependente precisa estar em coluna. Para exemplificar, podemos utilizar a variável death como se fosse o nosso desfecho de interesse e para isso necessitamos utilizar o parâmetro "by" e indicar o nome da variável dependente como argumento. Veja o código abaixo e como ficará a Tabela 19.5.

```
#Tabela 19.5
tbl_summary(trial2,
            label = list(trt ~ "Tratamento",
                         age ~ "Idade",
                         marker ~ "Marcador sérico",
                         death ~ "Óbitos"),
            by = death)
```

Tabela 19.5. Estabelecendo uma variável dependente (desfecho).

Characteristic	**Vivos**, N = 88[1]	**Óbitos**, N = 112[1]
Tratamento		
Droga A	46 (52%)	52 (46%)
Droga B	42 (48%)	60 (54%)
Idade	47 (36, 57)	48 (38, 58)
Unknown	2	9
Marcador sérico	0.73 (0.23, 1.33)	0.57 (0.20, 1.45)
Unknown	2	8

[1]n (%); Median (IQR)

As variáveis *age* (Idade) e *marker* (Marcador sérico) apresentam dados faltantes (*missing data*). Precisamos tratá-los, pois a função retorna "Unknown" para isso. Existe um parâmetro chamado "missing" que suporta três argumentos diferentes, de acordo com a nossa necessidade: "no",

caso você não deseje representar os dados faltantes na tabela; “ifany”, que é o argumento padrão, onde será mostrado dados faltantes caso a variável apresente pelo menos um dado faltante; e, “always”, onde irá mostrar sempre a frequência de dados faltantes, existindo ou não.

Um outro parâmetro diz respeito ao texto “Unknown”. Podemos alterá-lo para o que desejamos. Vamos demonstrar todos os valores faltantes de cada variável colocando o argumento “always" para o parâmetro “missing” e mudar “Unknown” para “Dados ausentes” (Tabela 19.6). Apesar de forçar neste exemplo a exibição dos dados faltantes, na maioria das vezes é melhor omitir essa informação para deixar a sua tabela mais limpa e objetiva. Como a maioria das tabelas têm os valores de n, então, subtende-se que haja valores faltantes quando a soma do n das classes não bate com o total de observações do estudo geral. Só exponha o quantitativo de dados faltantes se isso for realmente importante para o desenrolar da interpretação dos resultados.

```
#Tabela 19.6
tbl_summary(trial2,
            label = list(trt ~ "Tratamento",
                         age ~ "Idade",
                         marker ~ "Marcador sérico",
                         death ~ "Óbitos"),
            by = death,
            missing = 'always',
            missing_text = 'Dados ausentes')
```

Tabela 19.6. Tratando dados ausentes.

Characteristic	**Vivos**, N = 88[1]	**Óbitos**, N = 112[1]
Tratamento		
Droga A	46 (52%)	52 (46%)
Droga B	42 (48%)	60 (54%)
Dados ausentes	0	0
Idade	47 (36, 57)	48 (38, 58)
Dados ausentes	2	9
Marcador sérico	0.73 (0.23, 1.33)	0.57 (0.20, 1.45)
Dados ausentes	2	8

[1]n (%); Median (IQR)

Como a própria tabela demonstra, as variáveis contínuas foram representadas pela mediana e o intervalo interquartil. Entretanto, vimos em capítulos anteriores que se uma variável numérica apresentar distribuição normal, podemos utilizar a média e desvio padrão para representar descritivamente a distribuição desses dados. Por isso, imaginemos que a variável age (Idade) esteja na normal e precisamos utilizar média ao invés de mediana. Para isso, devemos modificar os

argumentos padrões do parâmetro "statistic". No Quadro 19.2 podemos ver os argumentos e o que será retornado na tabela construída, para as variáveis contínuas e categóricas, ou mesmo para ambos.

Quadro 19.2. Argumentos que podem ser utilizados no parâmetro "statistic" da função tbl_summary.

Argumentos	Descrição
Variáveis contínuas	
{median}	Mediana
{mean}	Média
{sd}	Desvio padrão
{var}	Variância
{min}	Valor mínimo
{max}	Valor máximo
{sum}	Soma
{p##}	Percentil, onde ## indica um inteiro entre 0 e 100
Variáveis categóricas	
{N}	Frequência da coluna
{n}	Frequência da célula
{p}	Porcentagem
Ambos tipos de variáveis, para dados faltantes	
{N_obs}	Total de observações
{N_miss}	Número de dados faltantes
{N_nonmiss}	Número de dados completos
{p_miss}	Porcentagem de dados faltantes
{p_nonmiss}	Porcentagem de dados completos

Vamos descrever a média e desvio padrão para as idades. Vamos aproveitar e remover a informação dos dados faltantes, enxugando a nossa tabela. Veja como fica em Tabela 19.7. Aqui especificamos a mudança de mediana para média apenas para a variável idade. Caso você queira modificar para todas as variáveis contínuas, troque no código abaixo "age" pela função all_continuous() e todas as variáveis numéricas serão representadas por média e desvio padrão.

```
#Tabela 19.7
tbl_summary(trial2,
            label = list(trt ~ "Tratamento",
                         age ~ "Idade",
                         marker ~ "Marcador sérico",
                         death ~ "Óbitos"),
            by = death,
            missing = 'no',
            statistic = list(age ~ "{mean} ({sd})"))
```

Tabela 19.7. Trocando a mediana e o intervalo interquartílico pela média e desvio padrão para a idade; removendo dados faltantes.

Characteristic	**Vivos**, N = 88[1]	**Óbitos**, N = 112[1]
Tratamento		
Droga A	46 (52%)	52 (46%)
Droga B	42 (48%)	60 (54%)
Idade	46 (15)	48 (14)
Marcador sérico	0.73 (0.23, 1.33)	0.57 (0.20, 1.45)

[1]n (%); Mean (SD); Median (IQR)

A nossa tabela está ficando cada vez melhor. Porém, precisamos cuidar de um detalhe que é importante para a visualização desses dados, que é a padronização da nossa tabela. O mais frequente quanto a padronização é a métrica que está sendo utilizada entre as variáveis e a quantidade de decimais após a vírgula desejo mostrar. Veja que idade e marcador sérico são variáveis contínuas, mas que estão com métricas de tendência central diferentes (respectivamente, média e mediana) e medidas de dispersão também diferentes (respectivamente, desvio padrão e desvio interquartílico). Além do mais, há valores na tabela que estão sem casas decimais (o que não é comum nas porcentagens) e outros valores têm duas casas decimais após a vírgula, o que deixa os valores visualmente bagunçados.

Primeiramente, vamos alterar a quantidade de casas após a vírgula para apenas uma, modificando o parâmetro "digits" dentro da função tbl_summary(), em uma lista. Após "digits" devemos especificar a ou as variáveis que desejamos alterar a quantidade de dígitos. Pode ser uma específica ou uma ou mais tipos de variáveis. Se quisermos modificar todas as variáveis contínuas, devemos estabelecer a função all_continuos(). Se o caso for as variáveis categóricas, devemos estabelecer a função all_categorical(). Veja como fica em Tabela 19.8.

```
#Tabela 19.8
tbl_summary(trial2,
            label = list(trt ~ "Tratamento",
                         age ~ "Idade",
                         marker ~ "Marcador sérico",
                         death ~ "Óbitos"),
            by = death,
            missing = 'no',
            statistic = list(age ~ "{mean} ({sd})"),
            digits = list(all_categorical() ~ 1,
                          all_continuous() ~ 1))
```

Tabela 19.8. Padronizando a quantidade de números após a vírgula.

Characteristic	**Vivos**, N = 88[1]	**Óbitos**, N = 112[1]
Tratamento		
Droga A	46.0 (52.3%)	52.0 (46.4%)
Droga B	42.0 (47.7%)	60.0 (53.6%)
Idade	46.1 (14.7)	48.2 (14.0)
Marcador sérico	0.7 (0.2, 1.3)	0.6 (0.2, 1.5)

[1]n (%); Mean (SD); Median (IQR)

Em segundo, vamos padronizar as variáveis contínuas para serem descritas apenas com a média de desvio padrão utilizando all_continuous(). Mas antes de rodar esse comando, vamos aprender sobre funções complementares básicas.

É interessante saber que o pacote gtsummary apresenta diversas função acessórias para a construção de tabela através da função tbl_summary(). Vamos demonstrar aqui algumas. Por exemplo, a função modify_header() servirá para modificar nomes das colunas, pois existe nela um parâmetro chamado "label" o qual atribuiremos o nome que queremos para a coluna *Characteristic*. Mudaremos para "Variáveis" e aproveitaremos e colocaremos esse nome em negrito usando asteriscos duplos. A segunda função acessória que usaremos será a add_n(), pois com ela adicionaremos os valores absolutos de observações por linha, o que é muito comum nas tabelas científicas. A terceira função será bold_labels(), a qual colocará o nome de cada variável linha em negrito. Por fim, usaremos a função add_p() para pôr na tabela o p-valor da estatística de comparação de grupos ou de associação entre variáveis (depende, claro, dos tipos das variáveis em jogo).

Sobre a função add_p(), ela retornará o resultado de um teste estatístico que muito provavelmente você não tenha desejado. A minha sugestão é que coloque essa função sempre que puder, mas modifique posteriormente os p-valores das análises que você realmente deseja. Se utilizar e for modificar o p-valor, lembre-se sempre de modificar a parte de notas da tabela para a análise que você realizou de fato.

Dessa forma, na tabela a seguir (Tabela 19.9) vamos padronizar as variáveis contínuas e adicionar as funções acessórias que foram comentadas acima. Veja como ficará:

```
#Tabela 19.9
tbl_summary(trial2,
            label = list(trt ~ "Tratamento",
                         age ~ "Idade",
                         marker ~ "Marcador sérico",
                         death ~ "Óbitos"),
            by = death,
            missing = 'no',
```

```
          statistic = list(all_continuous() ~ "{mean} ({sd})"),
          digits = list(all_categorical() ~ 1,
                        all_continuous() ~ 1)) %>%
  modify_header(label = "**Variáveis**") %>%
  add_n() %>%
  bold_labels() %>%
  add_p()
```

Tabela 19.9. Acrescentando funções acessórias.

Variáveis	**N**	**Vivos**, N = 88[1]	**Óbitos**, N = 112[1]	**p-value**[2]
Tratamento	200			0.4
Droga A		46.0 (52.3%)	52.0 (46.4%)	
Droga B		42.0 (47.7%)	60.0 (53.6%)	
Idade	189	46.1 (14.7)	48.2 (14.0)	0.5
Marcador sérico	190	1.0 (0.9)	0.9 (0.8)	0.6

[1]n (%); Mean (SD)
[2]Pearson's Chi-squared test; Wilcoxon rank sum test

Na Tabela 19.9 percebemos que não há mais nenhuma variável descrita como a mediana, pois usamos a função all_continuos() como argumento do parâmetro "statistic". Os testes estatísticos que a função add_p() realizou foram o teste de qui-quadrado de Pearson (entre as variáveis categóricas) e o teste de Wilcoxon (para a comparação de medianas entre os vivos e os óbitos). Isso está descrito na nota 2, logo abaixo da tabela e a minha sugestão recai sobre isso e repito: se não forem os testes estatísticos que você deseja, realize seus testes estatísticos individualmente e troque os resultas e a descrição da nota pelos que você realizou de fato.

Existem outras funções acessórias, como a add_ci() que retorna o intervalo de confiança, além de outros parâmetros da função tbl_summary() a serem modificados de acordo com a necessidade. Entre na documentação do pacote e explore outras possibilidades de personalização.

Por fim, vamos aprender como exportar essas tabelas de uma forma coerente e, principalmente, que seja compatível com arquivos de texto. Precisaremos instalar um pacote que irá nos ajudar no processo, chamado flextable.

```
install.packages('flextable')
library(flextable)
```

Com esse pacote poderemos salvar nossas tabelas em diversos formatos de arquivo: docx (Microsoft Word), html (que podem ser abertos em navegadores de internet), png (imagem), pptx (Wicrosoft Power Point) e rtf (do acrônimo de Rich Text Format, também mantido pela Microsoft como arquivo de texto). Cada formato tem a sua função própria para salvar a nossa tabela: save_as_docx(), save_as_html(), save_as_image(), save_as_pptx(), save_as_rtf(),

respectivamente. Devemos estabelecer o nome do arquivo (junto com o destino da pasta, caso você queira salvar em uma pasta específica) no parâmetro "path".

Como exemplo, iremos exportar a tabela até agora construída para um arquivo do Word. Então, utilizaremos save_as_docx(). Acredito que os formatos docx e pptx (para construir apresentações) sejam os formatos mais comuns e eu sugiro que você teste-os primeiramente. Caso dê algum problema de incompatibilidade salvando a tabela em docx, tente via rtf. Mas, evite a todo custo exportar as suas tabelas no formato de imagem. Primeiramente, imagens não serão facilmente remodeladas posteriormente (tente trocar um valor em uma tabela salva como imagem e entenderá o que digo); segundo, a qualidade de uma imagem dentro do seu texto não será a mesma a de um texto digitado; terceiro, imagens ocupam mais espaço do que texto no seu arquivo original.

Dito isso, vamos exportar nossa tabela escrevendo o código a seguir. Mas, antes precisaremos transformá-la em um objeto *flextable* para que as informações no R sejam corretamente transpostas para outros formatos. Para isso utilizaremos após o *pipe* (%>%) a função as_flex_table(). É simples. Veja como fica:

```
#Salvando Tabela 19.9 em arquivo Word
tbl_summary(trial2,
            label = list(trt ~ "Tratamento",
                         age ~ "Idade",
                         marker ~ "Marcador sérico",
                         death ~ "Óbitos"),
            by = death,
            missing = 'no',
            statistic = list(all_continuous() ~ "{mean} ({sd})"),
            digits = list(all_categorical() ~ 1,
                          all_continuous() ~ 1)) %>%
  modify_header(label = "**Variáveis**") %>%
  add_n() %>%
  bold_labels() %>%
  add_p() %>%
  as_flex_table() %>%
  save_as_docx(path = 'tabela1.docx')
```

Veja que o nome do arquivo exportado foi "tabela1.docx". Se você salvou o arquivo, porém não está encontrando-o, rode a função getwd() para saber em qual pasta o R está alocando os arquivos exportados. Aqui no meu computador está salvando neste caminho:

```
getwd()
[1] "C:/Users/lucianogama/Documents"
```

19.2 GRÁFICOS

Os gráficos são representações visuais de dados estatísticos, projetados para facilitar a compreensão, interpretação e análise de padrões, tendências e relações entre as variáveis. São ferramentas facilitadoras na demonstração do que realmente é importante a ser mostrado.

A importância dos gráficos na bioestatística é significativa por diversas razões:

- **Comunicação eficiente**: gráficos proporcionam uma maneira eficaz de demonstrar resultados de pesquisas. Resultados de tabelas são eficientes também, mas podem cansar o leitor. Gráficos podem ser atrativos para chamar a atenção do leitor ao que é mais importante dentre os resultados;
- **Facilitam a compreensão**: a visualização de dados por meio de gráficos torna mais fácil para as pessoas entenderem informações complexas;
- **Identificação de tendências**: é mais fácil identificar tendências, padrões e anomalias nos dados quando trabalhamos com séries temporais ou dados de regressão. Com uma tabela isso pode custar muitas linhas ou colunas, o que pode não ser tão atrativo;
- **Comparação visual**: gráficos facilitam visualmente a comparação de diferentes grupos. Uma barra ou linha mais elevada do que outra pode indicar um nível médio mais elevado ou de maior frequência.

Não confunda gráfico com figura. Uma figura é uma representação visual mais genérica, podendo ser uma foto, um exame de imagem, diagramas, fluxogramas, um mapa ou gráficos. Então, nos trabalhos científicos todo gráfico é uma figura, mas nem toda figura é um gráfico. Além disso, uma figura pode conter um ou mais gráficos ou a combinação com os outros elementos.

Existem diversos tipos de gráficos possíveis. A escolha do gráfico adequado está relacionada aos tipos de variáveis a serem utilizadas para a construção dele. Passaremos por várias confecções gráficas diferentes e é importante que você capte o tipo de cada variável utilizada em cada situação.

O R já vem nativamente com um pacote específico para a construção de gráficos chamado graphics. As funções `plot()`, `boxplot()` e `hist()` são as mais utilizadas desse pacote (que você não precisa nem instalar e abrir biblioteca). Os gráficos gerados são bons, mas não lá muito elegantes assim e são mais difíceis a customização. Entretanto, pode ser uma boa opção rápida para visualizar algum resultado exploratório.

Para exemplificar o uso básico dessas funções, vamos construir um objeto chamado de "x" com 50 números aleatórios. Depois, vamos pôr esse objeto como argumento nessas funções e observar o que será retornado na Figura 19.1.

```
#Figura 19.1
set.seed(2222)
x <- rnorm(50)
plot(x)
boxplot(x)
hist(x)
```

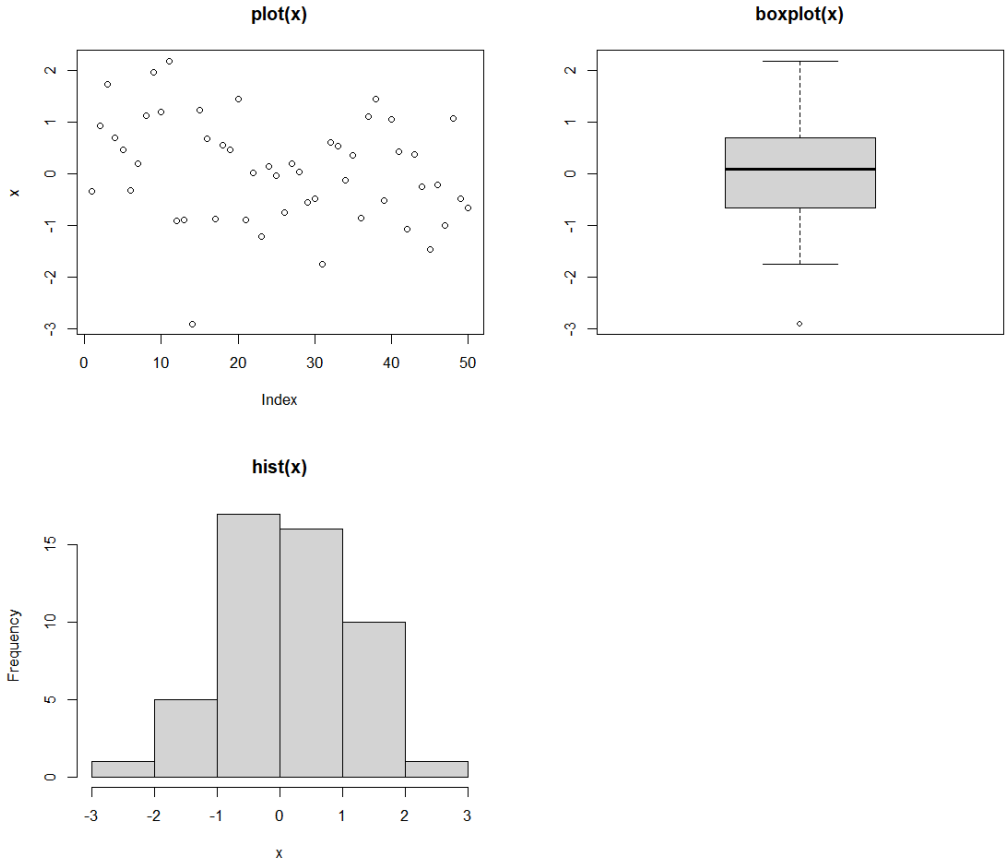


Figura 19.1 Gráficos construídos a partir do pacote graphics do R.

Particularmente eu gosto muito desses gráficos, mas eles são muito crus. Dificilmente eu os utilizo para compor meus relatórios e artigos científicos. Caso você deseje se aprofundar nas funções contidas no pacote graphics, acesse o site https://stat.ethz.ch/R-manual/R-devel/library/graphics/html/00Index.html ou busque na internet por "r package graphics" ou ao similar.

Existem opções mais elegantes de gráficos. Ao longo dessa final jornada utilizaremos o pacote ggplot2 para codificarmos os gráficos. Esse pacote já faz parte do megapacote tidyverse,

então, se você já instalou o tidyverse, basta abrir essa biblioteca. Caso ainda não tenha instalado, instale-o. Contudo, pode-se instalar e abrir o ggplot2 individualmente. Vaja as possibilidades:

```
install.packages('tidyverse')
library(tidyverse)

#ou

install.packages('ggplot2')
library(ggplot2)
```

A função ggplot() têm uma estrutura própria, ou seja, a sua própria sintaxe computacional. Algumas informações nós precisamos adicionar obrigatoriamente (o banco de dados, o tipo de gráfico e o seu "mapa"), enquanto que outras é opcional (escalas dos eixos, temas, facetas, etc.). Essa estrutura pode ser resumida da seguinte maneira:

```
ggplot(data = <banco de dados>) +                                    Obrigatório
<função_geom_>(mapping = aes(<mapa gráfico>),
stat = <estatística>, position = <posição> +
<funções de facetas> +                                               Opcional
<funções de escalas> +
<funções de temas>
```

No esquema, tudo que estiver entre < > deverá ser preenchido com algum argumento, sendo ele uma função, um objeto ou números indicadores de tamanho ou posição. Podemos ainda começar o nosso gráfico manipulando o banco de dados usando *pipe* (%>%) e só depois começar o esquema gráfico. Inclusive faremos isso com frequência em nossos exemplos. Isso é útil em diversas situações.

A função aes() também é obrigatória, a qual e responsável pela estética do gráfico, principalmente qual variável ficará no eixo x, qual no eixo y, qual ficará no preenchimento de cores, enfim. Ela pode ser especificada dentro de cada função geom_ ou ser especificada dentro de ggplot(). Quando especificada em ggplot() qualquer função geom_ terá a mesma estética, enquanto que quando especificada em geom_, cada função geom_ terá a sua estética específica. Então, fica a dica:

- Usar aes() em ggplot(): quando um ou mais gráficos sobrepostos apresentarem as mesmas variáveis nos mesmos eixos;
- Usar aes() em cada função geom_: quando cada gráfico sobreposto tiver variáveis diferentes nos eixos.

19.2.1 Histogramas

Um histograma é uma representação gráfica que ilustra a distribuição de dados numéricos, geralmente apresentada em barras verticais. Cada barra no histograma representa um intervalo (também chamados de *bins*) dessa variável numérica e o eixo y representa as frequências absolutas dentro de cada intervalo. Histogramas são frequentemente utilizados em bioestatística e análise de dados para facilitar a compreensão da distribuição e variação de um conjunto de dados. Devo lembrar que já trabalhamos ao longo do livro, quando discutimos sobre a normalidade dos dados numéricos.

Para exemplificar, vamos usar o conjunto de dados CO2 (atenção: letras em maiúsculo). É um conjunto de dados com 84 linhas e 5 colunas de um experimento sobre a tolerância ao frio da espécie de gramínea *Echinochloa crus-galli*. Vamos avaliar a distribuição da variável uptake, que é um vetor numérico das taxas de absorção de dióxido de carbono pela planta.

```
#Figura 19.2
CO2 %>%
  ggplot(aes(x = uptake)) +
  geom_histogram()
```

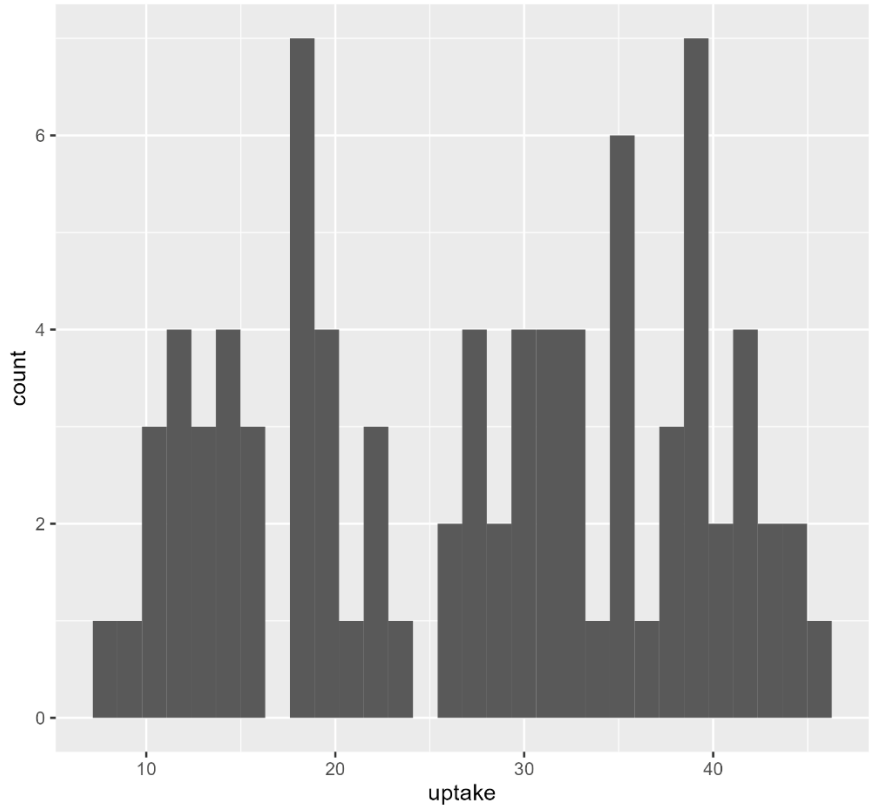


Figura 19.2. Histograma via ggplot.

Parâmetros úteis para a função geom_histogram():

- bin: valor numérico, mudará a quantidade de barras;
- binwidth: valor numérico, mudará o intervalo de cada barra para o eixo x, ou seja, barras mais longas ou mais curtas, você mudará aqui;
- fill: especifica a cor interna das barras;
- color: especifica a cor das cordas.

```
#Figura 19.3
#histograma apresentará 6 barras
CO2 %>%
  ggplot(aes(x = uptake)) +
  geom_histogram(bins = 6)
```

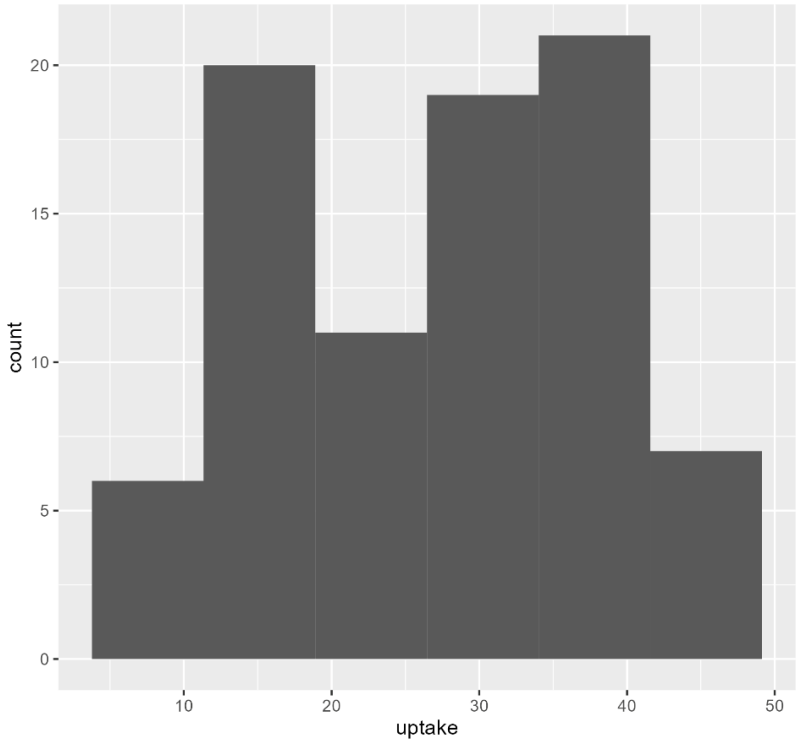


Figura 19.3. Histograma com seis barras.

```
#Figura 19.4
#o espectro de cada barra terá valor de 5
CO2 %>%
  ggplot(aes(x = uptake)) +
  geom_histogram(binwidth = 5)
```

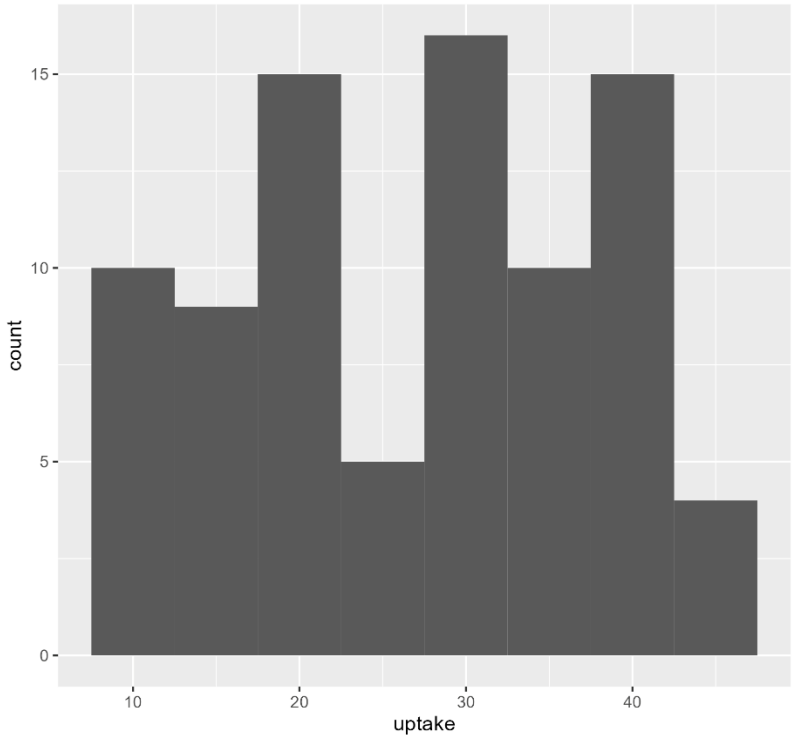


Figura 19.4. Histograma com espectro diferente do padrão.

```
#Figura 19.5
#mudando a cor das barras e suas bordas
CO2 %>%
  ggplot(aes(x = uptake)) +
  geom_histogram(fill = "white",
                 color = "black")
```

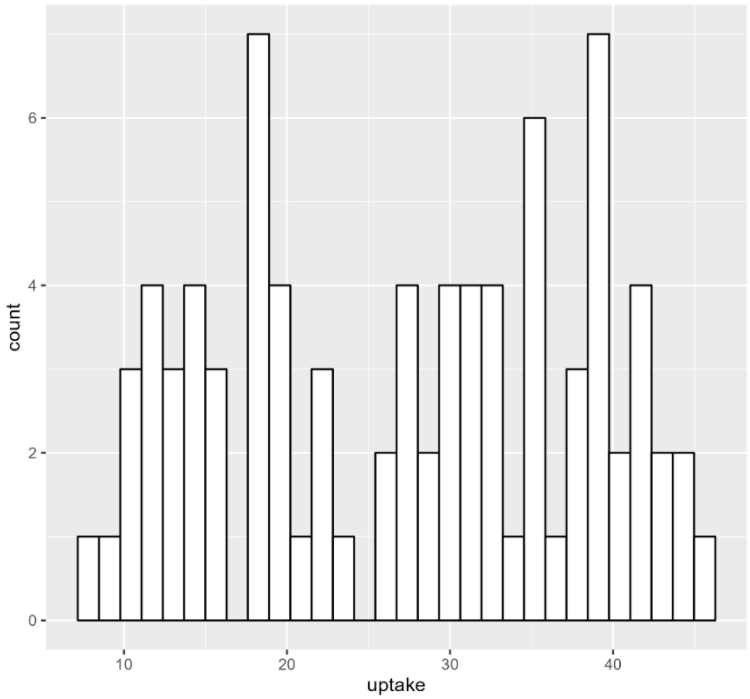


Figura 19.5. Histograma com cor modificada.

Algumas situações o histograma se mostra com picos bem diversos. Isso pode indicar que populações ou grupos diferentes estejam influenciando na variável numérica. Podemos verificar se isso ocorre pedindo para o ggplot2 separar os dados segundo uma variável categórica. Na bioestatística, geralmente isso pode ocorrer nas diferenças de sexo (homem x mulher), diferenças etárias (jovens x adultos), ou mesmo na presença da doença (caso x controle). No exemplo abaixo verificamos que há uma diferença da captura de CO_2 segundo o ambiente estar refrigerado (*chillie*) ou não (*nonchillie*) através da variável Treatment.

Utilizamos o parâmetro "position" como "identify" para que as barras não sejam sobrepostas entre as duas classes. O parâmetro "alpha" necessita de um valor numérico entre 0 e 1, que indicará a transparência das barras. Quanto mais próximo de 0, mas transparentes ficarão as barras. Na função scale_fill_manual() a gente especificará cores diferentes para cada classe.

```
#Figura 19.6
CO2 %>%
  ggplot(aes(x = uptake, fill = Treatment)) +
  geom_histogram(binwidth = 2,
                 position = "identity",
                 alpha = 0.7) +
  scale_fill_manual(values = c("chilled" = "blue",
                               "nonchilled" = "red"))
```

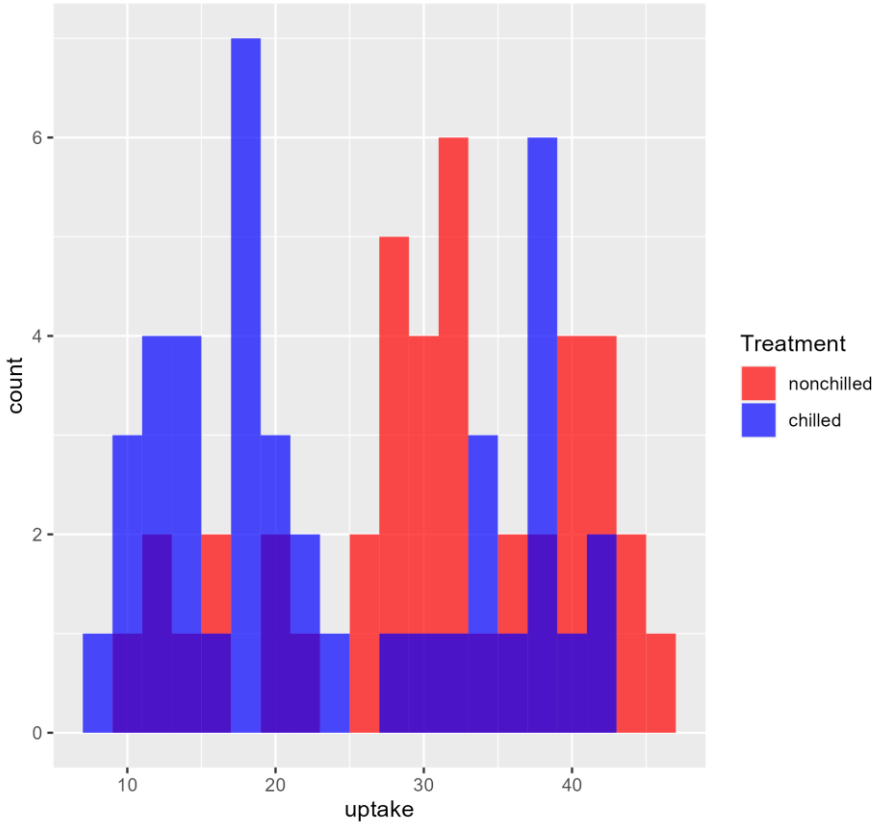


Figura 19.6. Histograma de populações diferentes.

19.2.2 Gráfico de densidade

O gráfico de densidade é muito semelhante ao histograma, contudo a suas linhas são suavizadas. Vamos utilizar os mesmos dados usados acima.

```
#Figura 19.7
CO2 %>%
  ggplot(aes(x = uptake)) +
  geom_density()
```

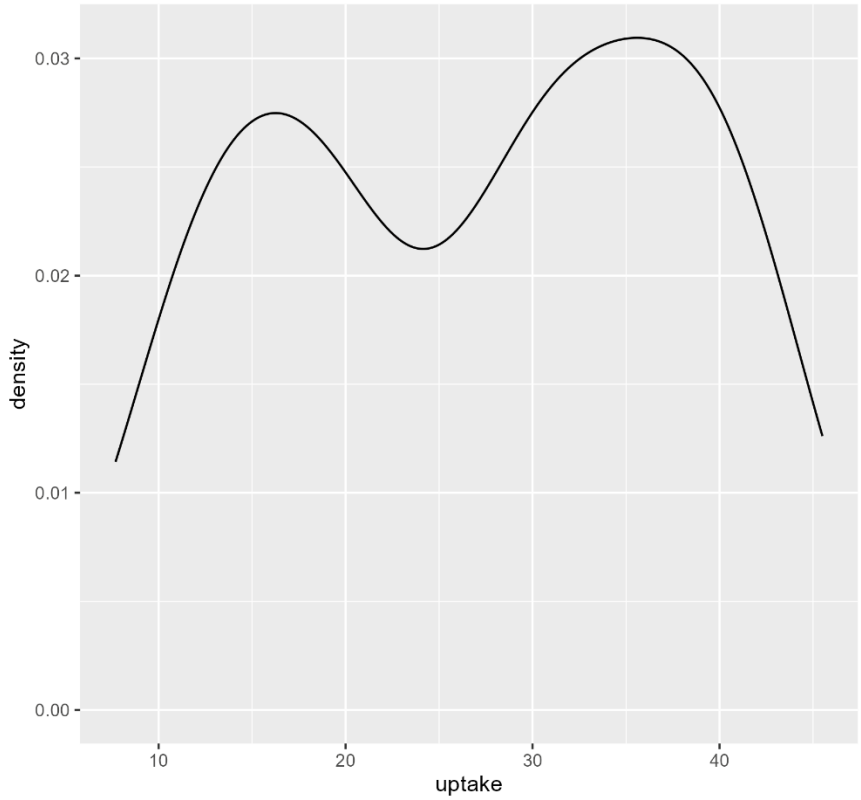


Figura 19.7. Gráfico de densidade.

Variando um pouco o gráfico, vamos preencher toda a área de azul com uma transparência mediana e remover a cor da borda.

```
#Figura 19.8
CO2 %>%
  ggplot(aes(x = uptake)) +
  geom_density(fill = 'blue',
               color = NA,
               alpha = 0.5)
```

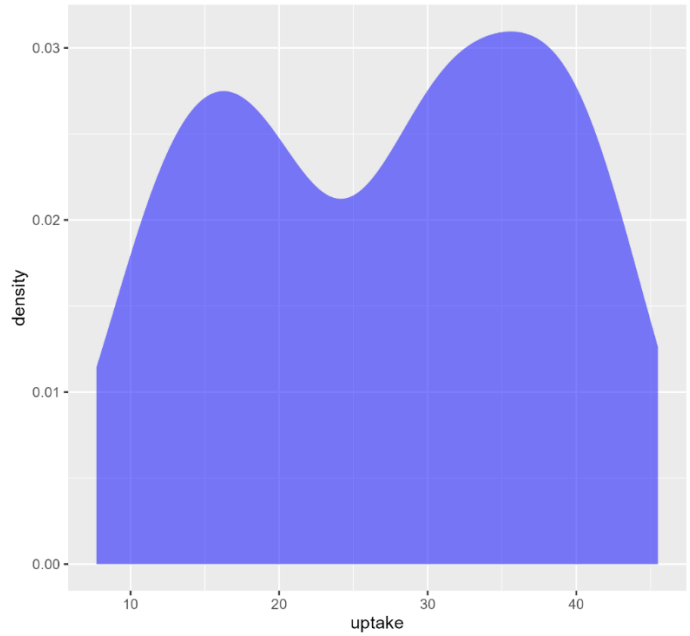


Figura 19.8. Gráfico de densidade preenchido e transparência de 0,5.

Podemos verificar como uma variável categórica influencia a variável numérica.

```
#Figura 19.9
CO2 %>%
  ggplot(aes(x = uptake, fill = Treatment)) +
  geom_density(alpha = 0.5) +
  scale_fill_manual(values = c("chilled" = "blue",
                               "nonchilled" = "red"))
```

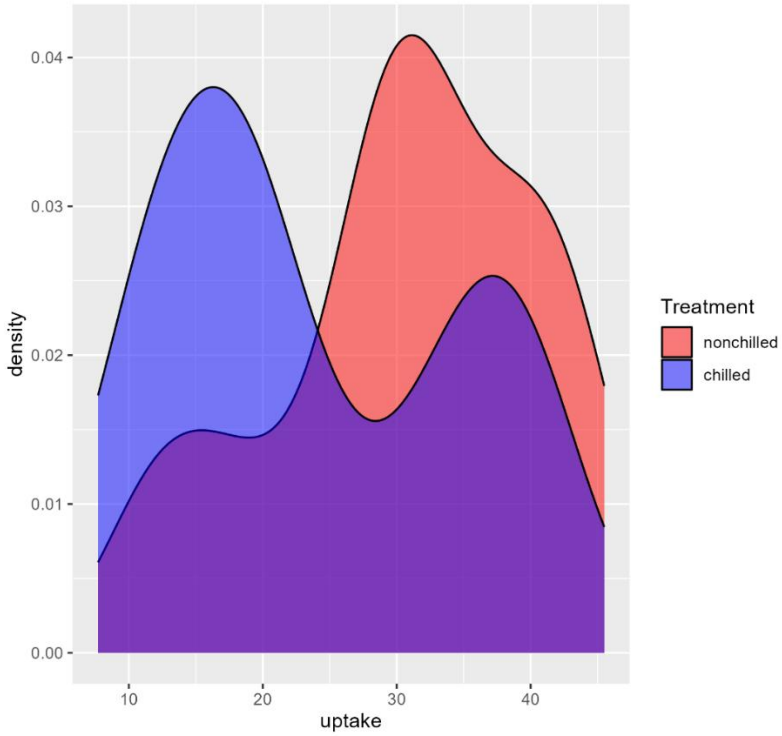


Figura 19.9. Gráfico de densidade de grupos diferentes.

19.2.3 Q-Q plot

O Q-Q plot, ou gráfico quantil-quantil, é uma ferramenta gráfica utilizada na análise estatística para avaliar se uma determinada amostra de dados numéricos segue uma distribuição normal. Nesse gráfico, os quantis observados da amostra são plotados contra os quantis esperados da distribuição teórica, geralmente a distribuição normal padrão. Se os pontos do Q-Q plot se aproximam de uma linha diagonal, isso sugere que a amostra segue a distribuição normal. Desvios sistemáticos ou padrões não lineares nos pontos podem indicar desvios da normalidade.

Usamos pontos para observar o quantis e a reta diagonal como referência. Por isso, usamos duas funções em conjunto: geom_qq() e geom_qq_line(). Vamos primeiro verificar como dados distribuídos na normal se comportam em um Q-Q plot.

```
#Figura 19.10
set.seed(1414)
dadosNormalizados <- data.frame(
  var = rnorm(100)
  )
dadosNormalizados %>%
  ggplot(aes(sample = var)) +
  geom_qq() +
  geom_qq_line()
```

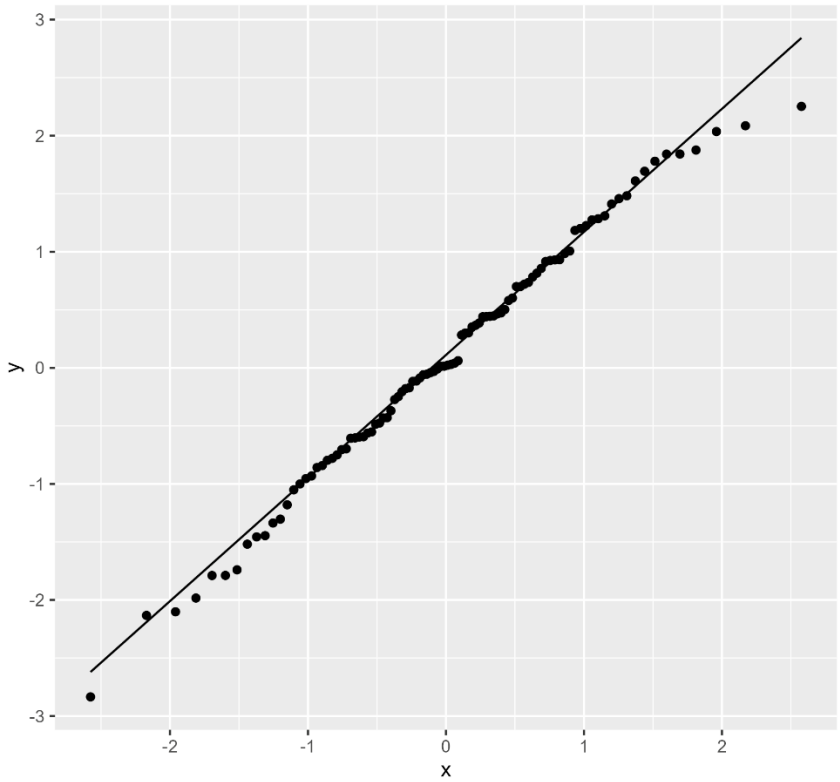


Figura 19.10. Q-Q plot demonstrando uma distribuição normal.

Quando os dados estão na distribuição norma, os pontos presentes no Q-Q plot ficam bem próximos à reta diagonal, como dito acima. Vamos nos certificar se a distribuição está mesmo na normal rodando o teste de Shapiro Wilk.

```
shapiro.test(dadosNormalizados$var)
      Shapiro-Wilk normality test

data:  dadosNormalizados$var
W = 0.9909, p-value = 0.737
```

Como o p-valor para teste não foi significativo, concluímos que realmente a distribuição segue a normal.

Vamos testar isso para os dados que já estávamos trabalhando anteriormente sobre captação de CO_2.

```
#Figura 19.11
CO2 %>%
  ggplot(aes(sample = uptake)) +
  geom_qq() +
  geom_qq_line()
```

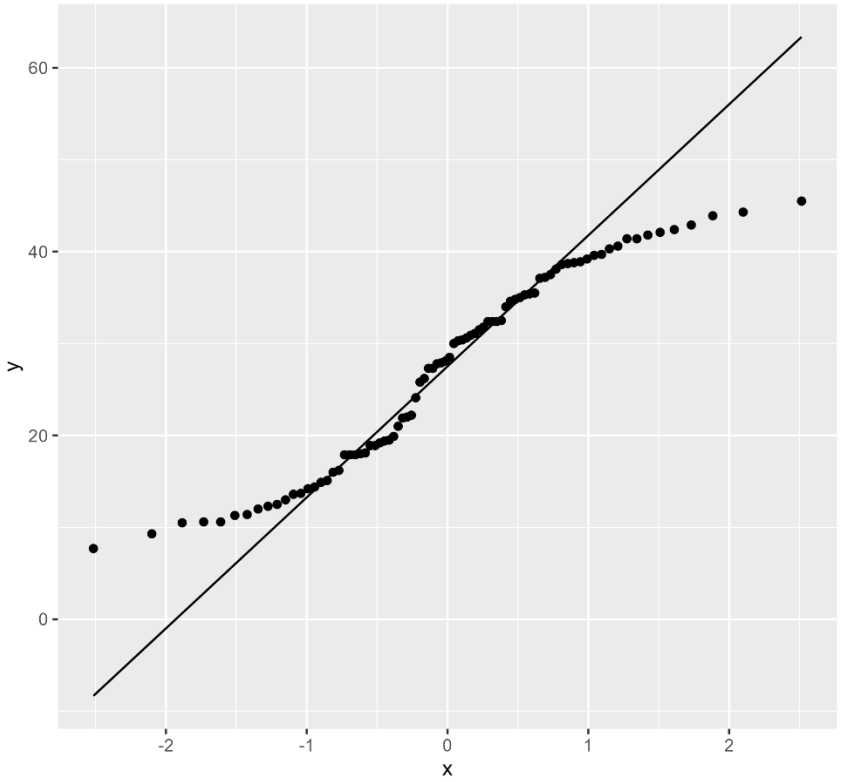


Figura 19.11. Q-Q plot demonstrando uma distribuição fora da normal.

Perceba o quão os pontos extremos fogem da linha de referência. Isso sugere que os dados não estão na distribuição normal. Vamos verificar a normalidade com o teste de Shapiro-Wilk novamente.

```
shapiro.test(CO2$uptake)

      Shapiro-Wilk normality test

data:  CO2$uptake
W = 0.94105, p-value = 0.0007908
```

Como o p-valor foi significativo ($P \leq 0,05$), concluímos que realmente os dados não estão na distribuição normal.

Geralmente usamos o Q-Q plot para entender a distribuição dos resíduos das regressões, pois o ideal é que o valores dos resíduos estejam na distribuição normal, caso contrário, é possível que as estimativas dos coeficientes não sejam tão boas assim, entre outros problemas preditivos. Veja exemplo de como avaliar os resíduos:

```
#Criando dados como exemplo
set.seed(111)
dados <- data.frame(
  resposta = rnorm(100),
  variavel1 = rnorm(100),
  variavel2 = rnorm(100)
)

#Modelo de regressão linear
modelo <- lm(resposta ~ variavel1 + variavel2, data = dados)

#Figura 19.12
#Q-Q plot com pacote graphics
plot(modelo, which = 2)
```

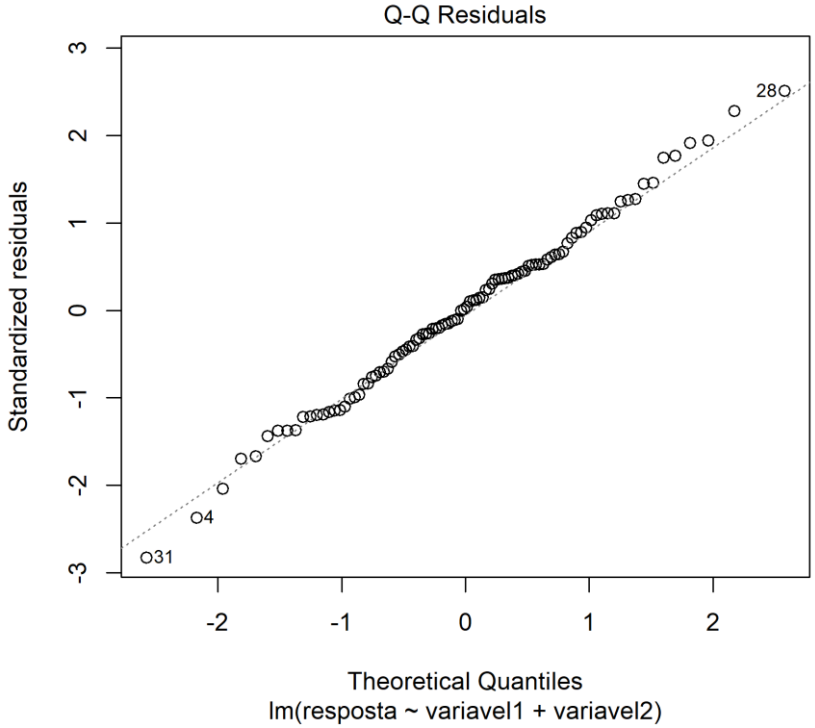


Figura 19.12. Q-Q plot com os resíduos de regressão linear.

```
#Figura 19.13
#Q-Q plot usando o ggplot2
modelo %>%
  ggplot(aes(sample = modelo$residuals)) +
  geom_qq() +
  geom_qq_line()
```

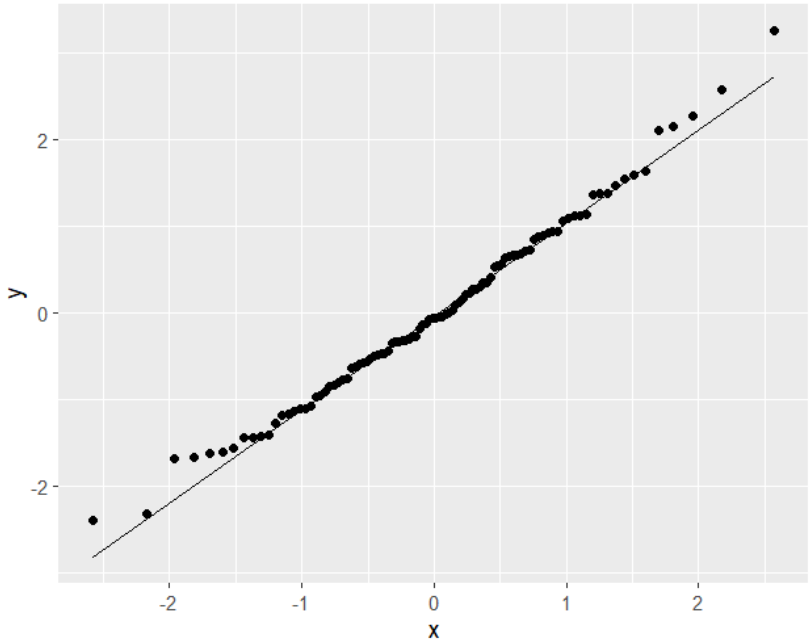


Figura 19.13. Q-Q plot com os resíduos de regressão linear via ggplot.

Perceba que o gráfico gerado pelo ggplot2 está mais "seco". Contudo, podemos modificar o título do gráfico, os títulos dos eixos e outras configurações as quais aprenderemos ao seguir do livro.

19.2.4 Gráfico de pontos

O gráfico de pontos, ou gráfico de dispersão, ou do inglês *scatterplot*, é uma representação visual que apresenta a relação entre duas variáveis numéricas através do uso de pontos em um plano cartesiano X e Y. O gráfico de dispersão acaba sendo uma ferramenta essencial na análise exploratória de dados, facilitando nas análises de regressão e na detecção de correlações, *outliers*, padrões e até agrupamentos.

Usando o ggplot2, a função relativa a geometria que iremos utilizar é geom_point(). Vamos usar um banco de dados chamado airquality, que apresenta dados sobre a qualidade de ar na cidade de Nova York. Vamos verificar como se comporta a relação entre a variável temperatura (dada em graus Fahrenheit) e a varável vento (originalmente dada em milhas por hora).

```
#Figura 19.14
airquality %>%
  ggplot(aes(x = Temp, y = Wind)) +
  geom_point()
```

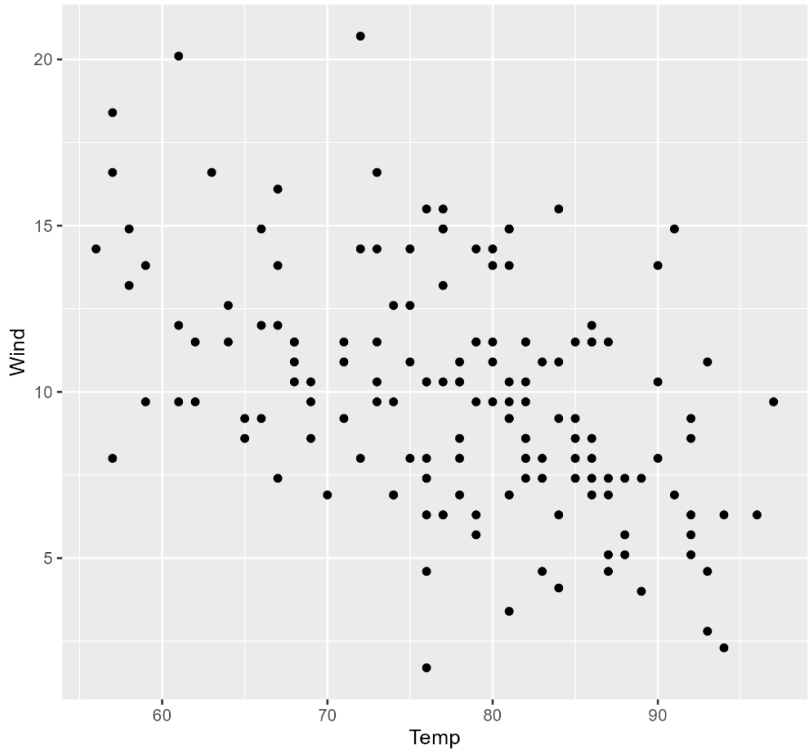


Figura 19.14. Gráfico de dispersão ou gráfico de pontos.

Parâmetros interessantes para geom_point():

- alpha: nível de transparência dos pontos;
- color: altera a cor dos pontos;
- shape: formato dos pontos;
- size: tamanho dos pontos.

Abaixo temos uma figura generalizando os tipos de pontos possíveis que podemos especificar no parâmetro "shape". Observe que números diferentes codificam os diferentes formatos. Então, em "shape" devemos atribuir um número.

Figura 19.15. Tipos de formas dos pontos possíveis.
Fonte: https://tidyverse.github.io/ggplot2-docs/articles/ggplot2-specs.html

Vamos alterar o gráfico feito anteriormente, mudando a cor dos pontos para vermelho, aumentar o tamanho para 3 e modificar a forma para quadrados.

```
#Figura 19.16
airquality %>%
  ggplot(aes(x = Temp, y = Wind)) +
  geom_point(color = 'red', size = 3, shape = 15)
```

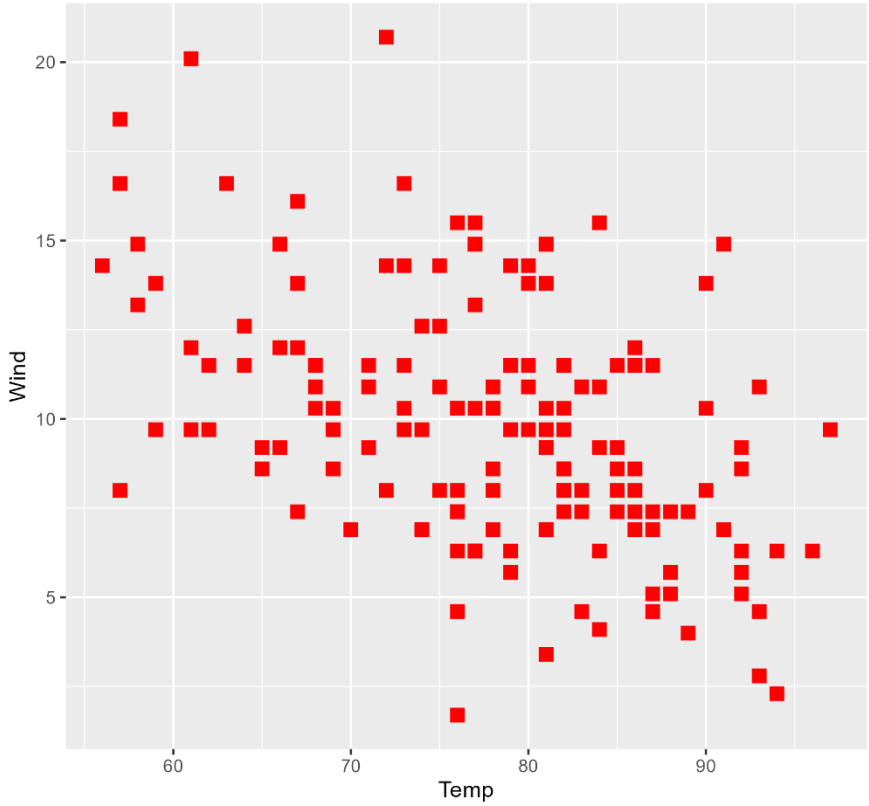


Figura 19.16. Gráfico de dispersão com "pontos" de forma quadrada.

19.2.5 Gráficos de tendência *smooth*

Usando duas variáveis contínuas, é possível observarmos a tendência dessa relação combinando três coisas: o gráfico de dispersão, minimamente uma regressão linear e a sua reta de tendência. Com a função geom_smooth() é possível calcular e criar rapidamente essa linha de tendência juntamente com o seu intervalo de confiança.

Vamos continuar usando os dados de temperatura e de velocidade dos ventos na cidade de Nova York para entendermos a relação entre tais variáveis, presentes no banco airquality.

```
#Figura 19.17
airquality %>%
  ggplot(aes(x = Temp, y = Wind)) +
  geom_point() +
  geom_smooth(method = "lm")
```

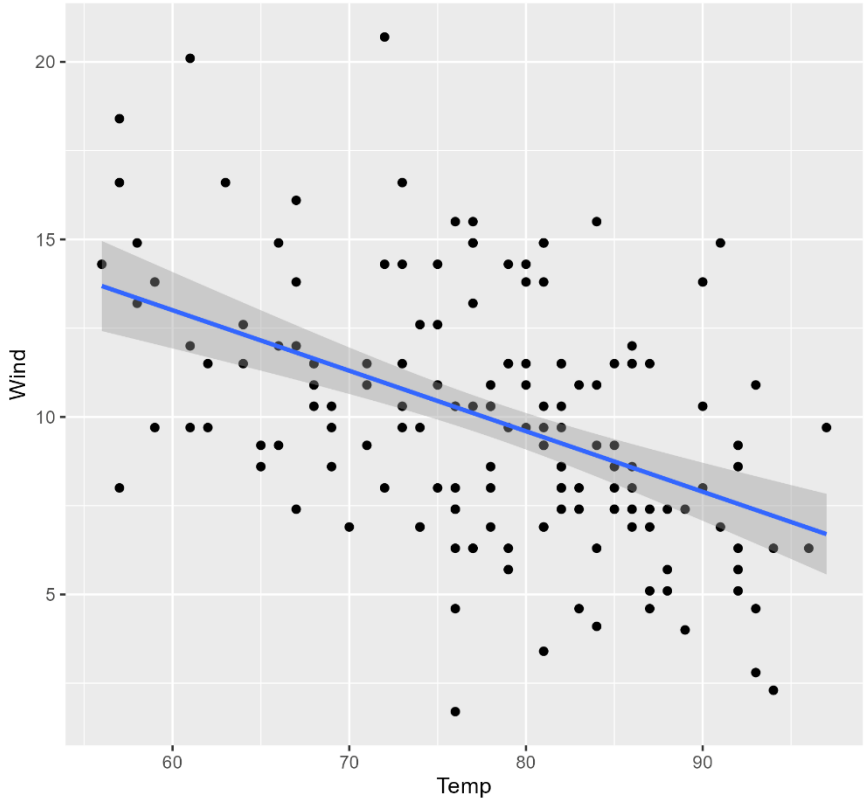


Figura 19.17. Gráficos de tendência *smooth* (linha azul).

A função geom_point() utilizada acima é opcional. Na omissão dela, apareceria apenas a linha de tendência azul. O espaço hachurado que cerca a linha é a região do intervalo de confiança da reta de regressão. A nível interpretativo, há uma tendência de quando se aumenta a velocidade do vento, diminui-se a temperatura, pois a inclinação da reta sugere variáveis inversamente proporcionais e provavelmente apresentam correção negativa.

Parâmetros interessantes da função geom_smooth():

- method: método de construção da linha de tendência;
    - lm: modelo de regressão linear
    - loess: modelo de regressão polinomial local;
    - glm: modelo linear generalizado;
    - gam: modelo aditivo generalizado.
- color: mudar a cor da reta;
- size ou linewidth: espessura da linha;

- se: booleano (TRUE/FALSE). Caso TRUE (padrão), a função adiciona o intervalo de confiança no gráfico; caso FALSE, o intervalo de confiança não aparecerá;
- linetype: tipo da linha.

Abaixo estão os tipos de linhas que podemos utilizar. A padrão é a linha cheia ou "solid". Outras mais comuns em uso é a tracejada ("dashed") e a pontilhada ("dotted").

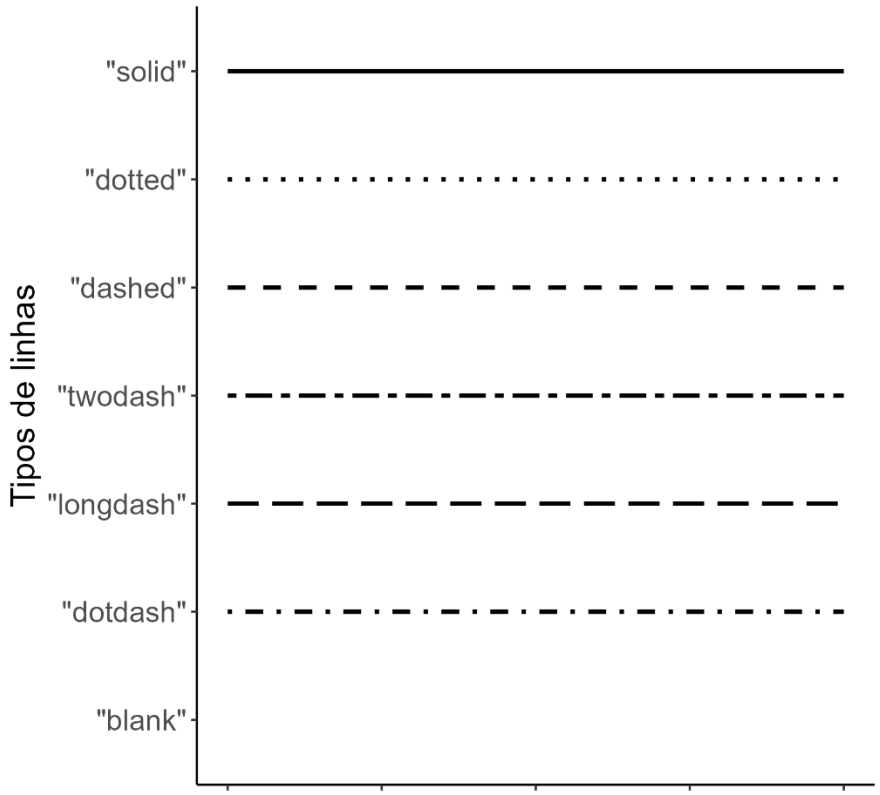


Figura 19.18. Tipos de linhas possíveis no ggplot.

Vamos exemplificar o refino no gráfico de tendência comparando o método linear ("lm") e o modelo de regressão polinomial local ("loess"). Vamos observar como cada um se comporta. Além disso, vamos pôr a linha de tendência do modelo polinomial de vermelho e tracejada. Usemos o banco iris.

```
#Figura 19.19
iris %>%
  ggplot(aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point() +
  geom_smooth(method = 'lm') +
  geom_smooth(method = 'loess',
              linetype = 'dashed',
              color = 'red')
```

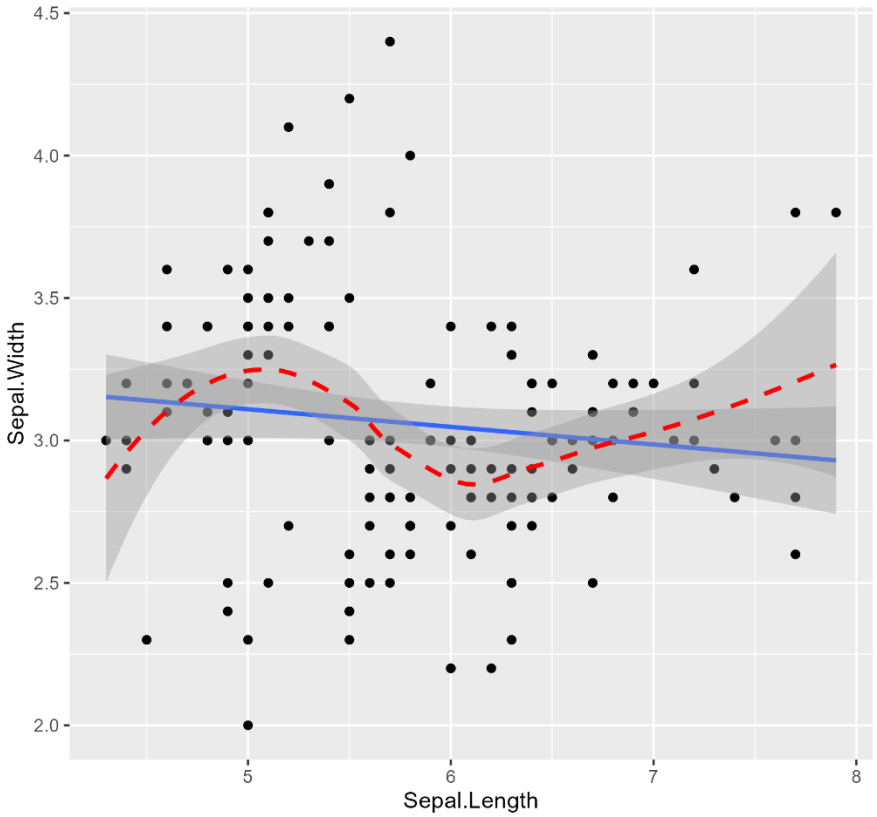


Figura 19.19. Linhas de tendência com métodos diferentes de tendência (linear e polinomial local).

19.2.6 Gráficos de linhas

Um gráfico de linha é um tipo de representação gráfica que utiliza linhas para conectar pontos de dados, exibindo a relação contínua entre duas variáveis ao longo de um eixo horizontal ou X. Normalmente, o eixo vertical ou Y representa a variável dependente. Dessa forma, a variável Y costumeiramente é a mais importante, enquanto que a variável X é a que dá ideia de período. Por isso, esse tipo de gráfico é comumente utilizado para visualizar tendências, padrões e mudanças ao longo do tempo. Por isso, esse tipo de gráfico é básico nos estudos de séries temporais.

Não confunda o gráfico de linhas com o gráfico de tendência *smooth*, pois esse último é gerado a partir de um modelo estatístico, enquanto que o gráfico de linhas é feito a partir dos dados diretamente. Por isso, usamos outra função: `geom_line()`.

Para exemplificar, vamos voltar a utilizar o banco airquality. Se você observar, ele verifica periodicamente certas características do ar. A variável *Month* indica o mês, enquanto que a variável

Day indica o dia da verificação. Vamos, portanto, filtrar o mês 5 (maio) e verificar a série de aferições do fluxo de vento pela variável *Wind* diariamente.

```
#Figura 19.20
airquality %>%
  filter(Month == 5) %>%
  ggplot(aes(x = Day, y = Wind)) +
  geom_line()
```

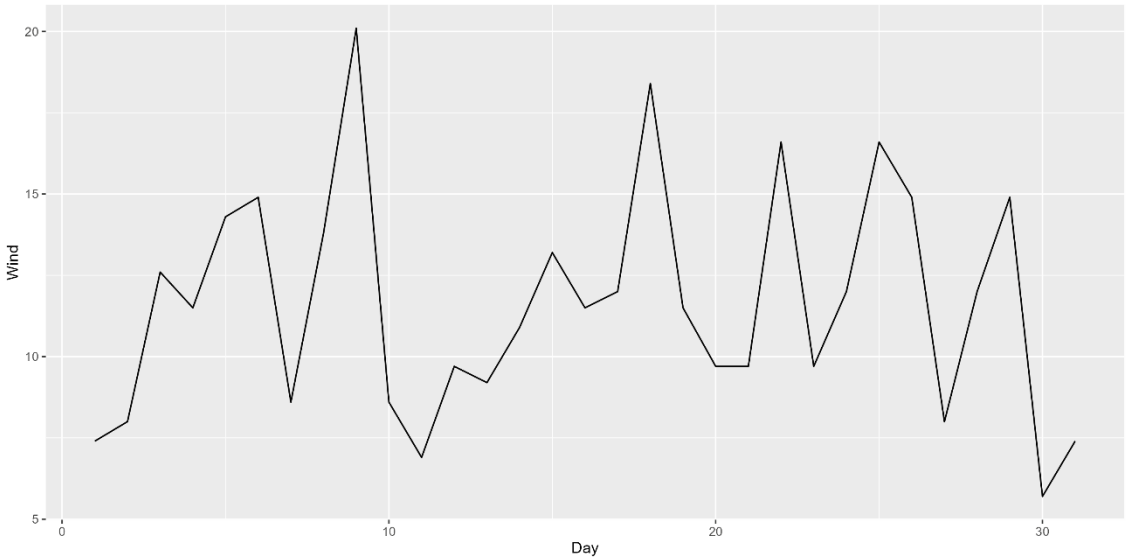


Figura 19.20. Gráfico de linhas simples.

Parâmetros interessantes da função geom_line():

- color: mudar a cor da reta;
- size ou linewidth: espessura da linha;
- linetype: tipo da linha;
- alpha: transparência.

Veja a mudança do gráfico:

```
#Figura 19.21
airquality %>%
  filter(Month == 5) %>%
  ggplot(aes(x = Day, y = Wind)) +
  geom_line(linewidth = 4,
            alpha = 0.3,
            color = '#4287f5')
```

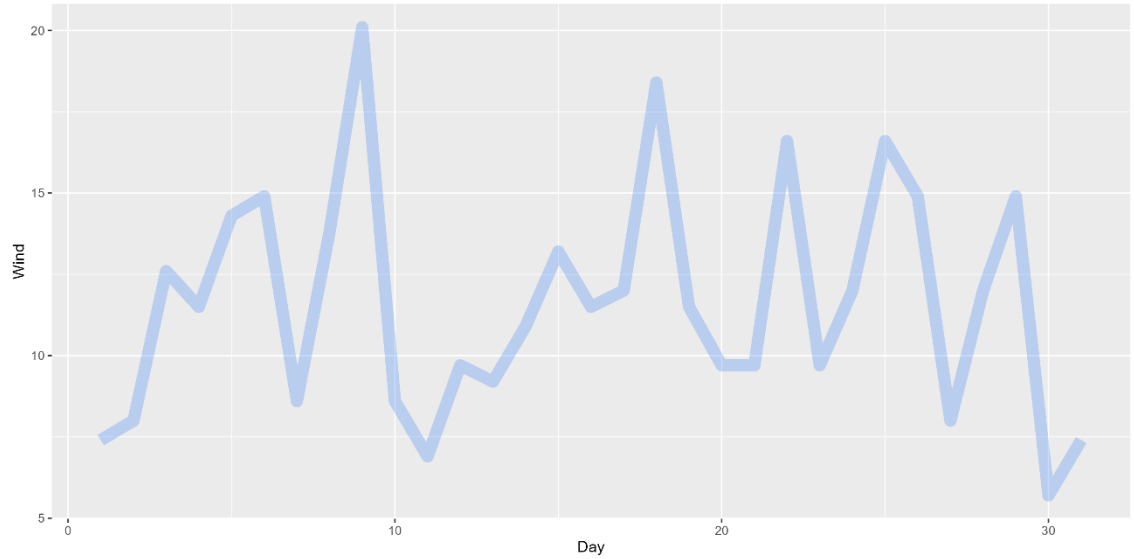


Figura 19.21. Gráfico de linhas com parâmetros de linha modificados.

Perceba que a linha ficou mais robusta, houve transparência de 30% e cor adicionada foi um azul claro escrita em hexadecimal. Mais à frente falaremos mais sobre cores, mas já quero lhes mostrar que há uma possibilidade diferente além de escrever o nome da cor em inglês.

19.2.7 Gráficos de área

O gráfico de área é muito similar ao gráfico de linhas, contudo, a intenção dele é hachurar todo a área que consta abaixo da linha até tocar o eixo y. Falamos em capítulos anteriores sobre a curva ROC e a sua área sob a curva (AUC). A curva ROC é basicamente um gráfico de linhas, mas quando calculamos a área que ele comporta, teremos o valor de AUC, que veio de um gráfico de área.

O ggplot2 tem a sua função específica para esse tipo de gráfico, a geom_area(). A nível comparativo, veja abaixo como ficaria um gráfico de área usando os mesmos dados utilizados para gerar o gráfico de linhas acima:

```
#Figura 19.22
airquality %>%
  filter(Month == 5) %>%
  ggplot(aes(x = Day, y = Wind)) +
  geom_area()
```

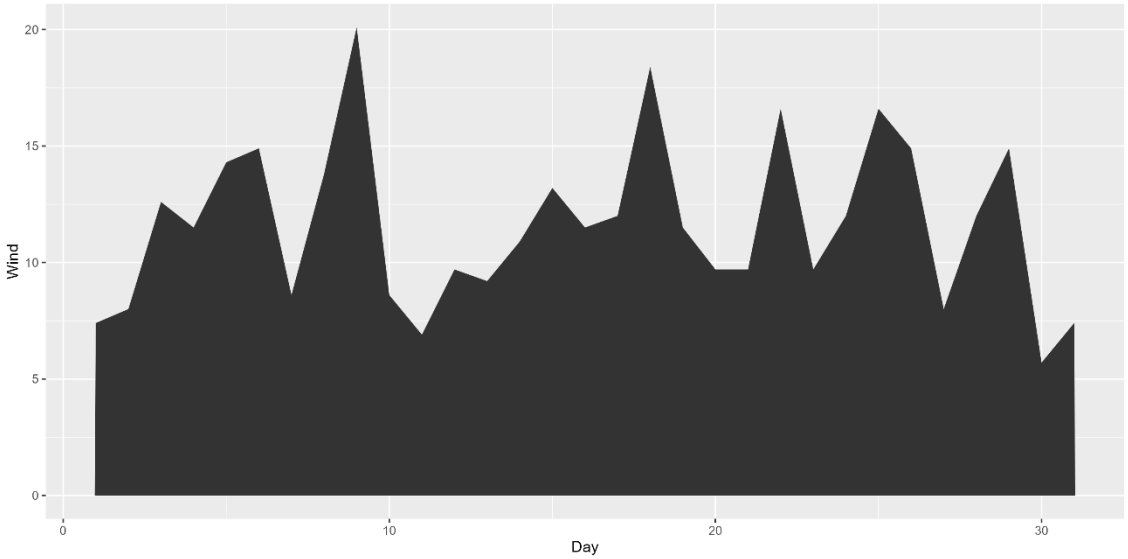


Figura 19.22. Gráfico de área.

Você pode usar os vários parâmetros que já trabalhamos em outras funções para mudar o tipo de linha e a sua espessura. Mas vamos mudar os dois parâmetros que são mais comuns: a transparência (modificando "alpha") e a cor de preenchimento (modificando o parâmetro "fill". Veja:

```
#Figura 19.23
airquality %>%
  filter(Month == 5) %>%
  ggplot(aes(x = Day, y = Wind)) +
  geom_area(fill = 'red',
            alpha = 0.4)
```

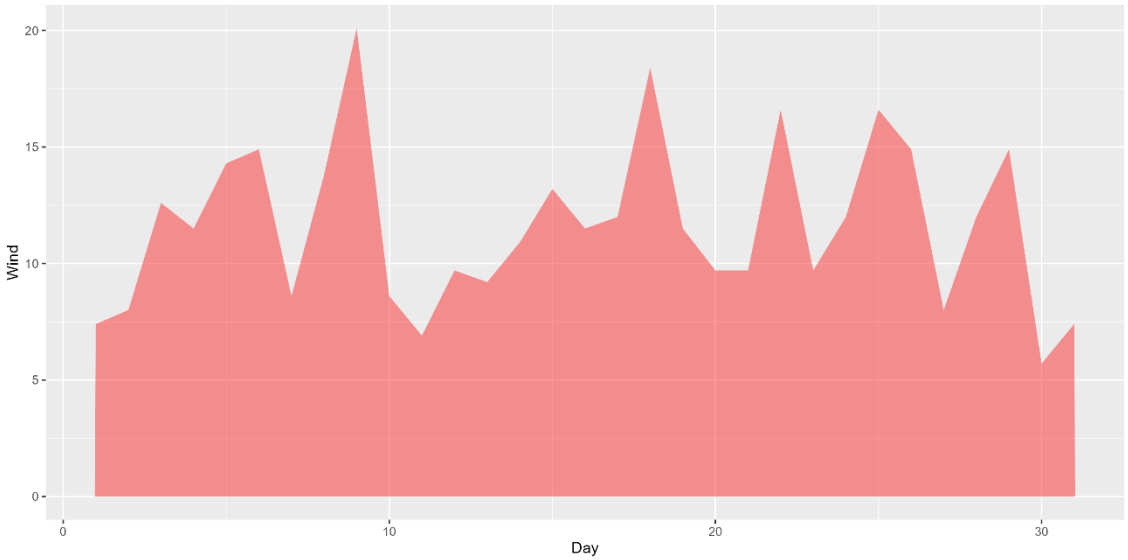


Figura 19.23. Gráfico de área com parâmetros de preenchimento modificados.

19.2.8 Gráfico de barras

No R, usando o pacote ggplot2, tanto geom_bar() quanto geom_col() são usados para criar gráficos de barras, mas há uma diferença crucial entre eles relacionada à maneira como os dados são mapeados para as alturas das barras.

1. geom_bar(): é usada quando você deseja contabilizar a frequência de uma variável categórica ou discreta e criar um gráfico de barras a partir dessas contagens. No exemplo trabalharemos com o banco chamado *storms*, que é o rastreamento de furacões ocorrido no Atlântico. Nesse banco há diversas variáveis, mas no exemplo verificaremos as frequências das classificações dos furacões com a variável *status*. Exemplo:

```
#Figura 19.24
data("storms")
storms %>%
#juntando dados por nome do furacão
  distinct(name, .keep_all = T) %>%
  ggplot(aes(x = status)) +
  geom_bar()
```

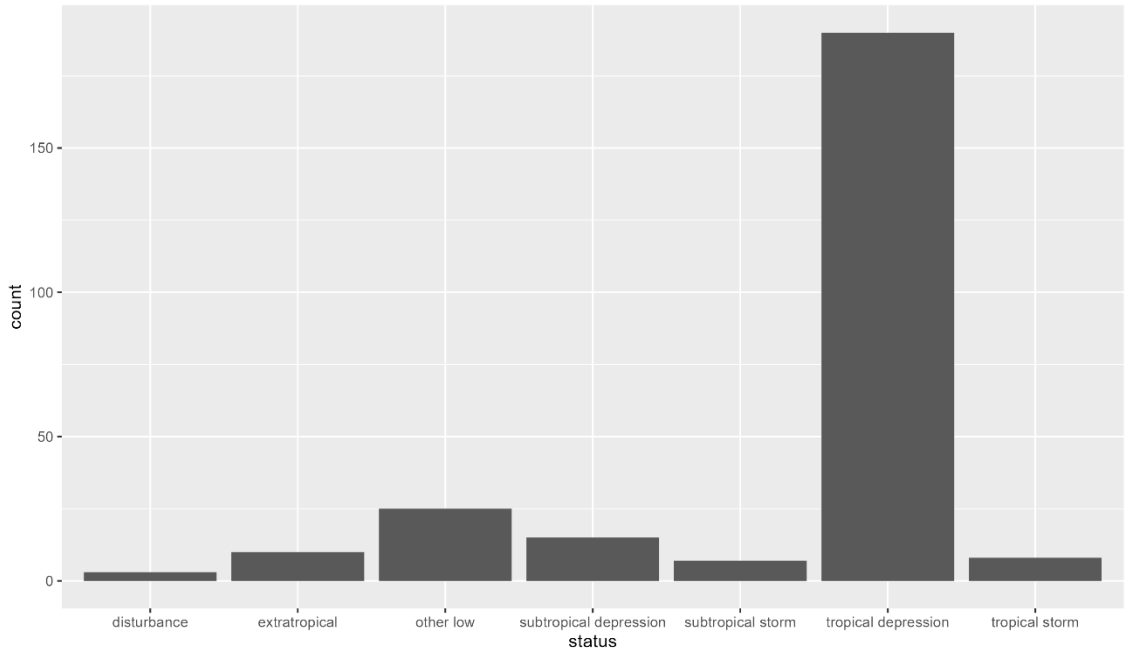


Figura 19.24. Gráfico de barras usando geom_bar().

2. geom_col(): é usada quando você já tem as contagens ou alturas das barras previamente contabilizadas (pode ser médias, medianas, frequências brutas, frequências relativas, enfim) e quer criar um gráfico de barras diretamente com esses valores. Ou seja, em vez de contar as ocorrências, geom_col() usa os valores fornecidos.

No exemplo usaremos o banco USArrests, que contém o levantamento sobre detenções por 100.000 residentes por agressão, homicídio e violação em cada um dos 50 estados dos EUA em 1973. Vamos avaliar graficamente a frequência de assaltos em cada estado. Como o nome dos estados estão na nomeação das linhas, vamos usar a função rownames_to_column() para construir uma variável nova com esses nomes. Como são muitos estados, vamos colocar as barras na horizontal, assim ficará melhor para visualizar, colocando no eixo y os estados e x a frequência de assaltos em cada um.

```
#Figura 19.25
data("USArrests")
USArrests %>%
  rownames_to_column(var = 'estados') %>%
  ggplot(aes(x = Assault, y = estados)) +
  geom_col()
```

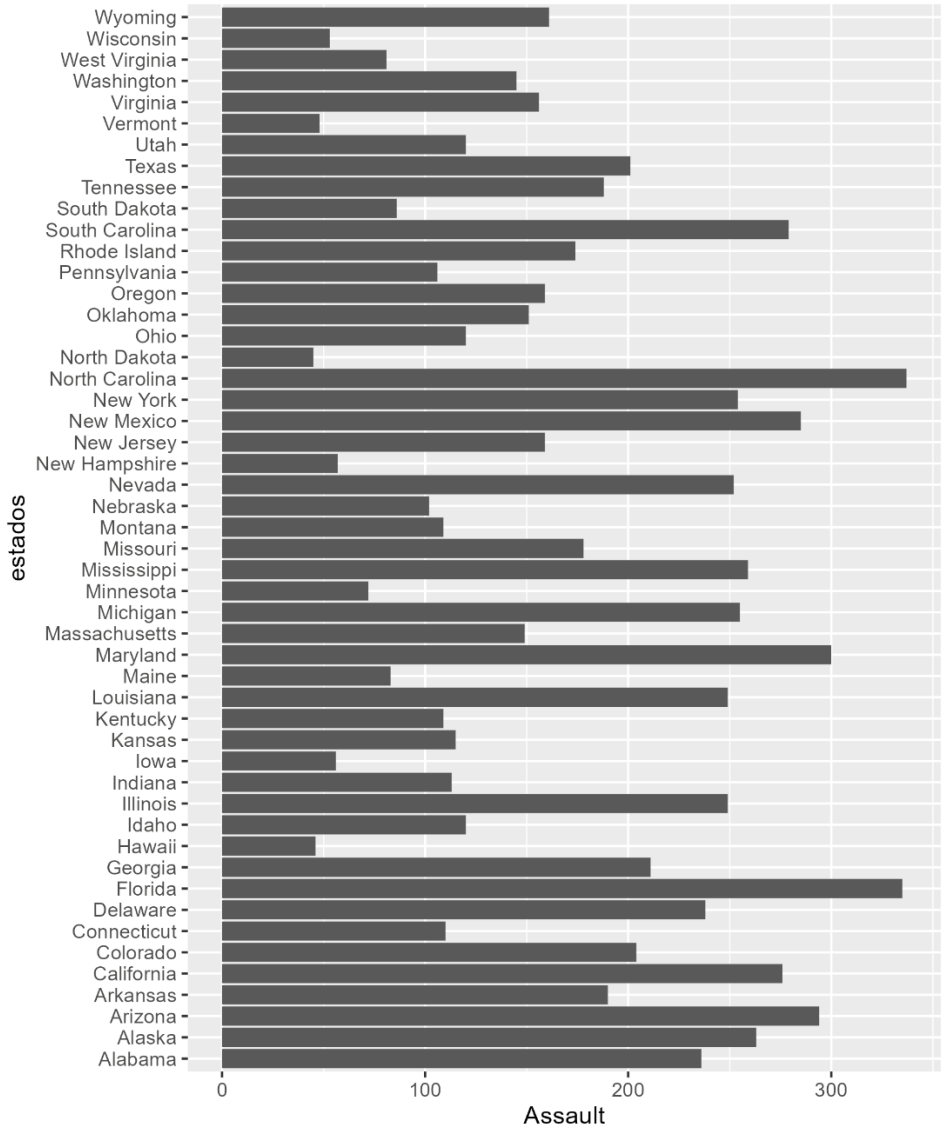


Figura 19.25. Gráfico de barras com geom_col(), posto na horizontal pela quantidade de classes.

Podemos modelar esses dados de acordo com a ordem das frequências (ou valores de grandeza para as classes da variável categórica). Para isso, podemos utilizar a função reorder(). Boa parte das vezes o mais importante é visualizar qual classe é mais frequente do que as demais e ordenar as classes acaba sendo uma ótima opção. Vaja como fica:

```
#Figura 19.26
USArrests %>%
  rownames_to_column(var = 'estados') %>%
  ggplot(aes(x = Assault, y = reorder(estados, Assault))) +
  geom_col()
```

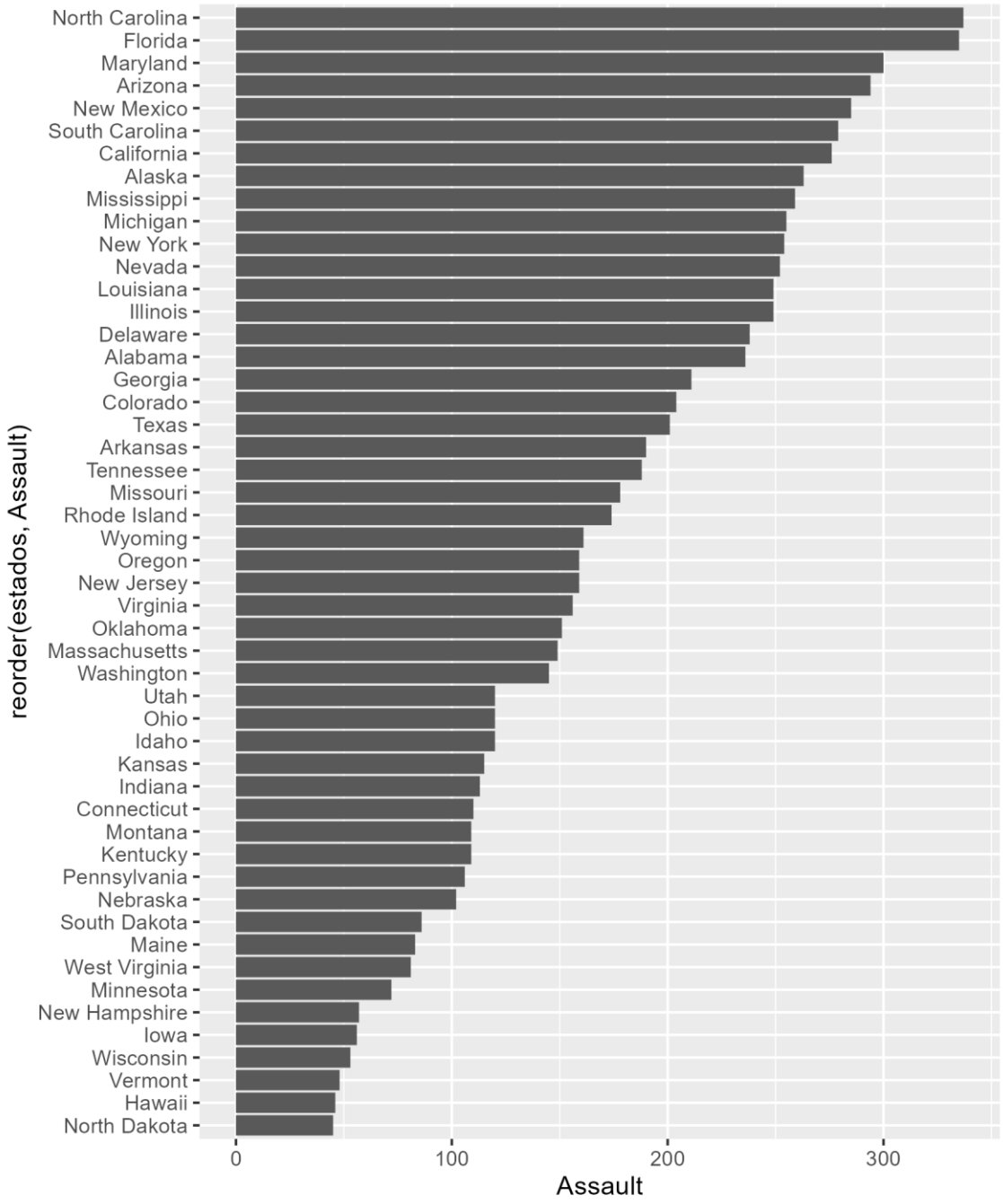


Figura 19.26. Gráfico de barras com classes ordenadas.

19.2.9 Diagrama de caixa ou *boxplot*

Já estudamos sobre o conceito e o uso do *boxplot* em capítulos iniciais. É importante relembrar que os diagramas de caixa utilizam o primeiro e terceiro quartis e no meio a mediana para representar a caixa. Por isso, quando houver teste de comparação de medias, talvez o *boxplot* não seja o gráfico mais adequado, porque por padrão a média não está demonstrada nele. Para representação da média use gráficos de barras com seu desvio padrão (ou o erro padrão da média).

Se a variável numérica estiver fora da distribuição normal, então o *boxplot* é muito bem-vindo. Se você desejar utilizar o *boxplot* quando a variável numérica estiver na normal, sugiro expressar a média no gráfico em uma tonalidade diferente da mediana (pode-se fazer isso com as funções de seguimento de reta, que veremos mais à frente).

Para confeccionarmos um *boxplot* usando o ggplot2, a função que iremos utilizar será a geom_boxplot(). Em nosso exemplo, vamos voltar a usar o banco *storms* e verificar como fica a distribuição da variável *wind* (velocidade máxima sustentada do vento em nós).

```
#Figura 19.27
storms %>%
  ggplot(aes(y = wind)) +
  geom_boxplot()
```

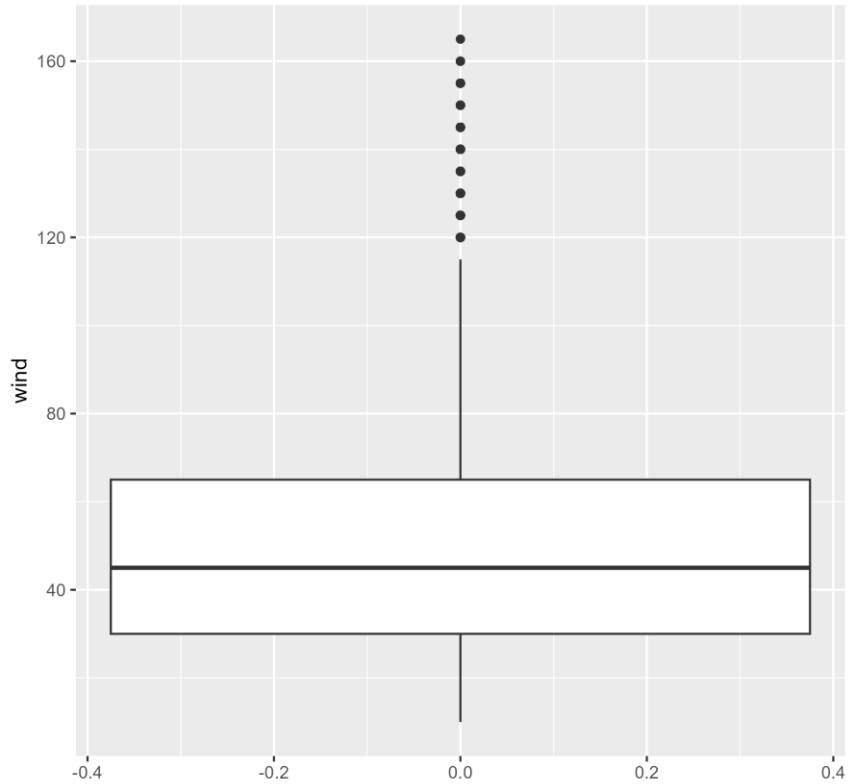


Figura 19.27. Diagrama de caixa ou *boxplot* simples.

Como podemos verificar, a mediana da velocidade máxima dos ventos é pouco mais de 40 nós. Contudo, vamos mais além. Nesse mesmo banco existe uma variável que categoriza as tempestades, chamada de *category*. Originalmente ela está como uma variável discreta e por isso vamos mudar para fator para que o R entenda que é uma categoria, não um numeral. Além disso, vamos remover os valores faltantes de *category*, pois não nos interessa mensurar a distribuição dos dados de velocidade de vento onde não há categorização da tempestade. Para isso usaremos a função na.omit().

```
#Figura 19.28
storms %>%
  na.omit(category) %>%
  ggplot(aes(x = as.factor(category), y = wind)) +
  geom_boxplot()
```

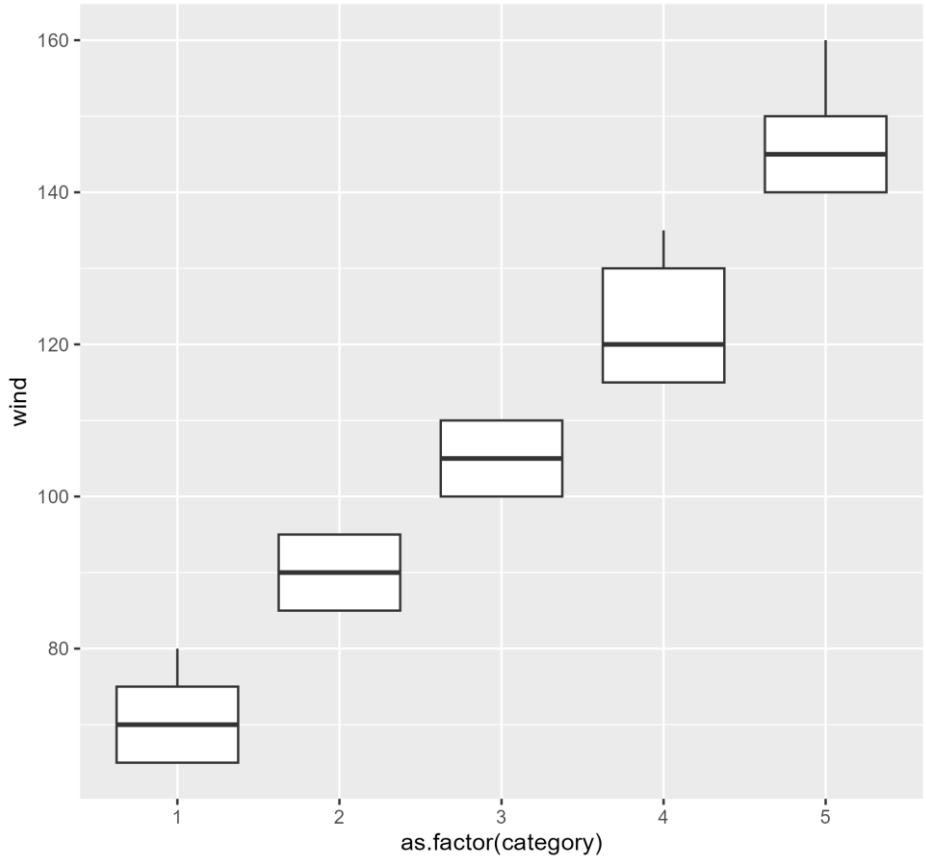


Figura 19.28. *Boxplots* de classes diferentes.

Pela leitura do gráfico, está óbvio que quanto maior é a classificação das tempestades, maior é a velocidade máxima que ela terá. Claro que para afirmarmos isso com mais segurança devemos realizar testes de comparação de médias e avaliar o valor de P.

Um tipo de gráfico interessante usando *boxplot* é quando adicionamos uma variável categórica para modificar cores de *boxplot* diferentes. Usando ainda parte do comando acima, vamos criar uma nova variável que classifique a latitude (lat), colocando "Abaixo" para latitudes que fiquem abaixo do Trópico de Câncer e "Acima" para latitudes que se encontram acima do Trópico de Câncer, tomando como ponto de corte 23,5, que é mais ou menos a latitude do referido trópico. Perceba que estarão em jogo três variáveis agora. Essa última especificaremos em "fill" (preenchimento). Se você especificar em color, mudará apenas as bordas dos *boxplot*. Com essa especificação adicional, criará no gráfico uma legenda lateral. Ademais, especificaremos as cores de preenchimento que desejamos manualmente utilizando na função scale_fill_manual(). Veja como fica:

```
#Figura 19.29
#criando nova variável categórica
storms$lat_class <- ifelse(storms$lat <= 23.5, "Abaixo", "Acima")
storms %>%
  na.omit(category) %>%
  ggplot(aes(x = as.factor(category), y = wind, fill = lat_class)) +
  geom_boxplot() +
  scale_fill_manual(values = c('Abaixo' = 'white',
                               'Acima' = 'gray40'))
```

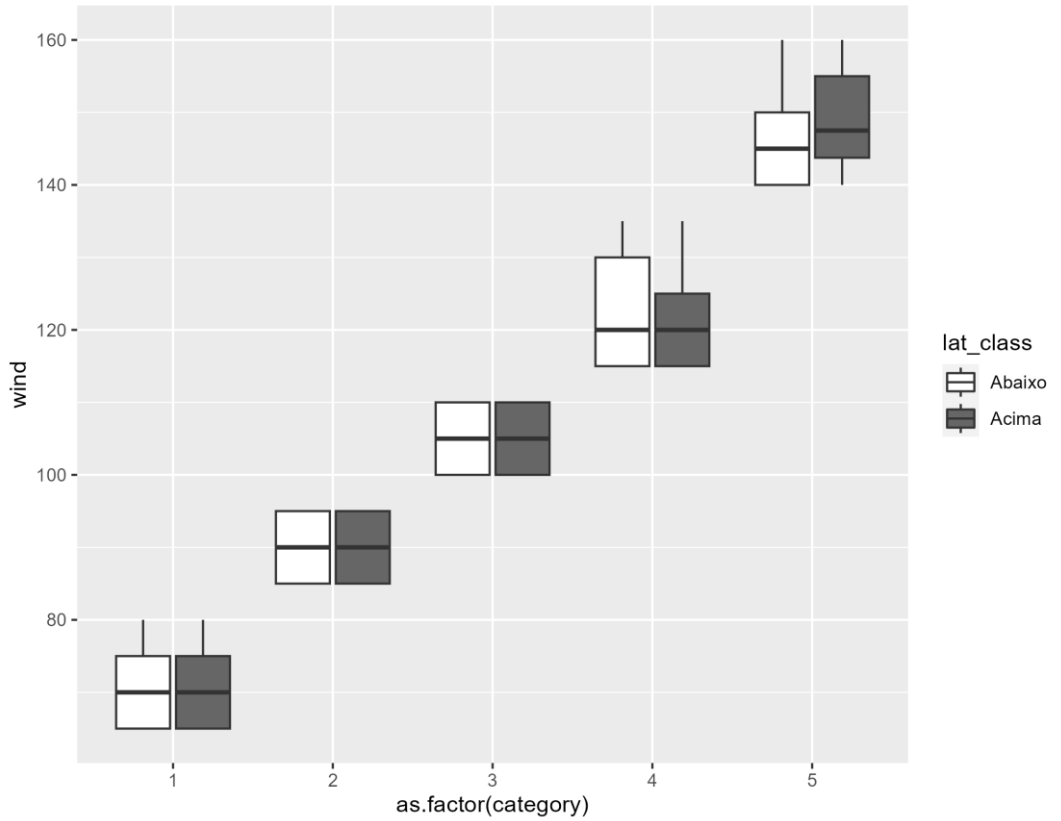


Figura 19.29. *Boxplots* de classes diferentes preenchidos por grupos distintos.

19.2.10 Gráfico de pontos sobrepostos ou *jittered points*

Esse tipo de gráfico, que você pode ser chamado também como *beeswarm plot*, tem finalidade parecida com o do *boxplot*, contudo, sua visualização é através de pontos. Gosto de dizer que é uma mistura de *boxplot* com do diagrama de dispersão. Para tramalharmos com o *jittered points* usaremos a função geom_jitter(). Nesse exemplo trabalharemos com o banco iris e compararemos a distribuição do comprimento da sépala das três espécies de iris (*Iris setosa*, *Iris virginica* e *Iris versicolor*).

```
#Figura 19.30
iris %>%
  ggplot(aes(x = Species, y = Sepal.Length)) +
    geom_jitter()
```

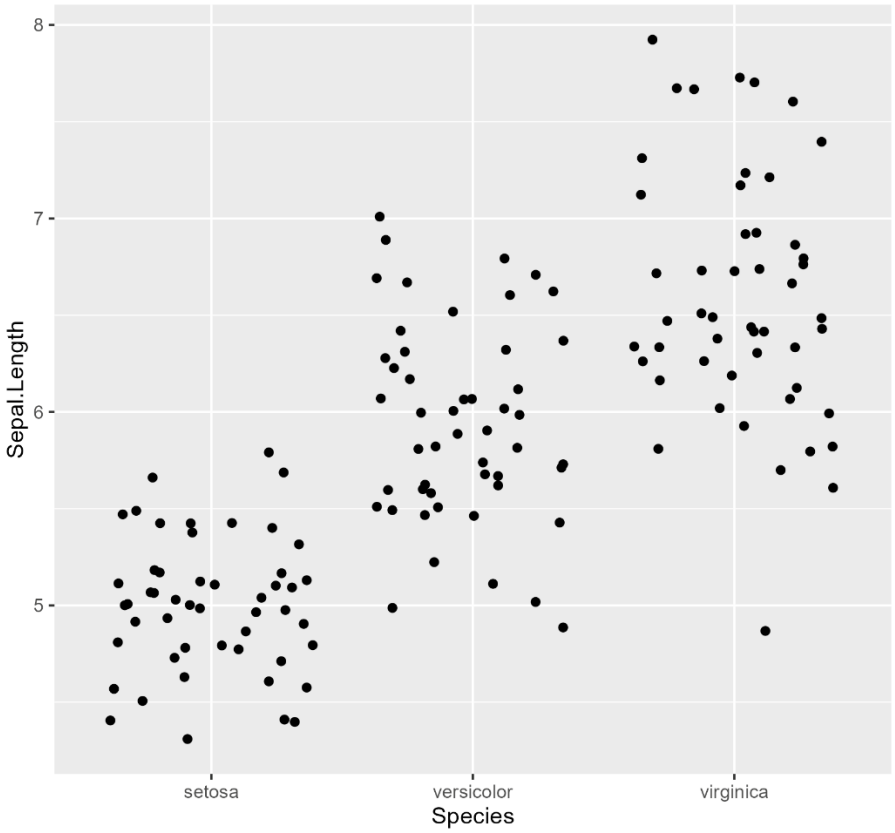


Figura 19.30. Gráfico de pontos sobrepostos ou *jittered points*.

Perceba que é muito parecido com um gráfico de pontos. A diferença é que no eixo x há uma variável categórica estabelecendo as espécies da planta e o ponto pertencente à espécie pode

variar levemente de lugar horizontalmente. Isso ajuda a ter uma noção de volume dos dados representados pelos pontos próximos. Veja abaixo a diferença entre usar geom_point() e geom_jitter() para o mesmo banco de dados. Com o gráfico de pontos, dados parecidos podem geras pontos sobrepostos e assim perdemos a ideia de volume de dados pertencente à classe, oque dificilmente ocorre com *jittered points*.

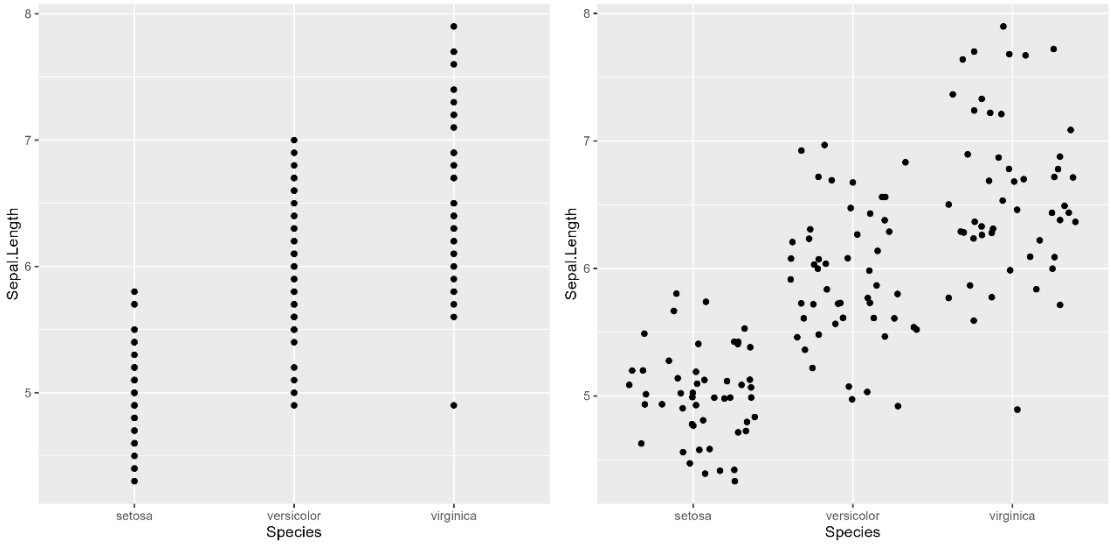


Figura 19.31. Diferença de gráfico de pontos por classe e *jittered points.*

Os parâmetros de importância são os mesmos do gráfico de pontos. Você pode mudar a cor, o tamanho, a forma, enfim. Além dos já conhecidos, o parâmetro "width" (largura) pode ajudar a melhorar o gráfico em algumas situações. Ele ajustará a distância dos pontos entre cada classe. Sugiro ajustar sempre entre 0,1 a 0,5. Quanto menor o valor, mas separados os pontos irão estar entre as classes.

19.2.11 Gráfico violino

Os gráficos violinos é uma representação visual que combina elementos dos *boxplots* e gráficos de densidade. Ele fornece uma visualização mais detalhada da distribuição de dados ao longo de uma variável contínua.

Para exemplificar o uso desse tipo de gráfico vamos utilizar o banco chamado ToothGrowth. Esse *dataset* refere-se a pesquisa que testou a resposta do comprimento dos odontoblastos (células responsáveis pelo crescimento dos dentes) em 60 porquinhos-da-índia. Cada animal recebeu um dos três níveis de dose de vitamina C (0,5, 1 e 2 mg/dia) por um dos dois métodos de administração: suco de laranja (*oranje juice* - OJ) ou ácido ascórbico (vitamina C - VC). Portanto,

nele há três variáveis: len (numérica; comprimento do dente), supp (fator; tipo de suplemento: VC ou OJ) e dose (numérica; dose de vitamina C administrada).

A função para construirmos o gráfico violino é geom_violin(). Vamos utilizar a variável dose como fator no eixo x e adicionalmente vamos mudar a cor de cada dose em tons de cinza diferentes usando a mesma variável em "fill". No eixo y colocaremos a variável len, de comprimento dental. As cores especificaremos em scale_fill_manual().

```
#Figura 19.32
data("ToothGrowth")
ggplot(ToothGrowth, aes(x=as.factor(dose), y=len, fill = as.factor(dose))) +
  geom_violin(trim=FALSE) +
  scale_fill_manual(values = c('gray70','gray50','gray30'))
```

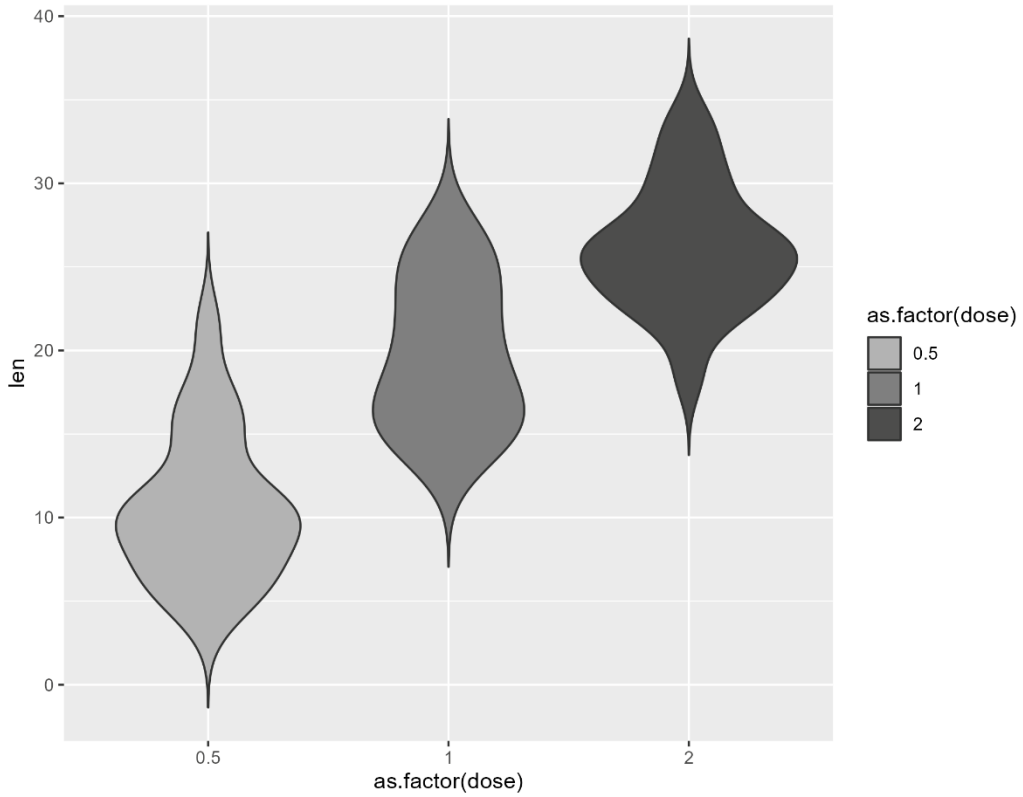


Figura 19.32. Gráfico violino por grupos.

Usamos o parâmetro "trim" como FALSE para afilar as pontas dos gráficos. São parâmetros interessantes para serem usados em geom_violin():

- trim: booleano: se TRUE (padrão), as extremidades do gráfico serão achatadas.
- draw_quantiles: estabelecer o quantis que deseja pôr no gráfico. A variação vai de 0 a 1. O valor 0.25 é o primeiro quartil, 0.5 o segundo quartil (mediana), 0.75 o terceiro quartil.
- color: cor de borda.

- fill = preenchimento de todos os gráficos.

No exemplo abaixo vamos pôr os quartis no gráfico.

```
#Figura 19.33
ggplot(ToothGrowth, aes(x=as.factor(dose), y=len, fill = as.factor(dose))) +
  geom_violin(trim=FALSE, draw_quantiles = c(0.25, 0.5, 0.75)) +
  scale_fill_manual(values = c('gray70','gray50','gray30'))
```

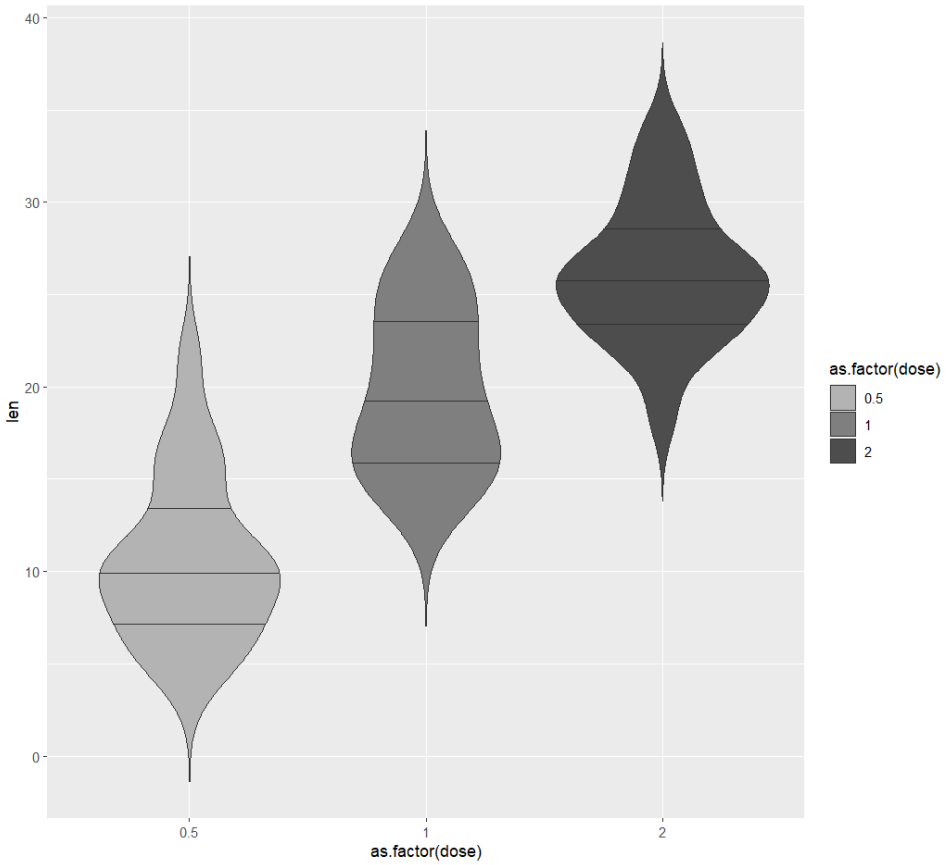


Figura 19.33. Gráfico violino com quartis.

Já no exemplo abaixo, nós vamos calcular as médias para cada dose e coloca-las no gráfico como linhas vermelhas horizontais em cada violino, usando geom_segment().

```
#Figura 19.34
#Calculando médias
aggregate(len ~ as.factor(dose), data = ToothGrowth, mean)

ggplot(ToothGrowth, aes(x=as.factor(dose), y=len, fill = as.factor(dose))) +
  geom_violin(trim=FALSE) +
  scale_fill_manual(values = c('gray70','gray50','gray30')) +
  geom_segment(aes(x = 0.5, xend = 1.5,
                   y = 10.605, yend = 10.605),
                   color = 'red', linewidth = 1) +
  geom_segment(aes(x = 1.5, xend = 2.5,
                   y = 19.735, yend = 19.735),
               color = 'red', linewidth = 1) +
```

```
  geom_segment(aes(x = 2.5, xend = 3.5,
                   y = 26.1, yend = 26.1),
               color = 'red', linewidth = 1)
```

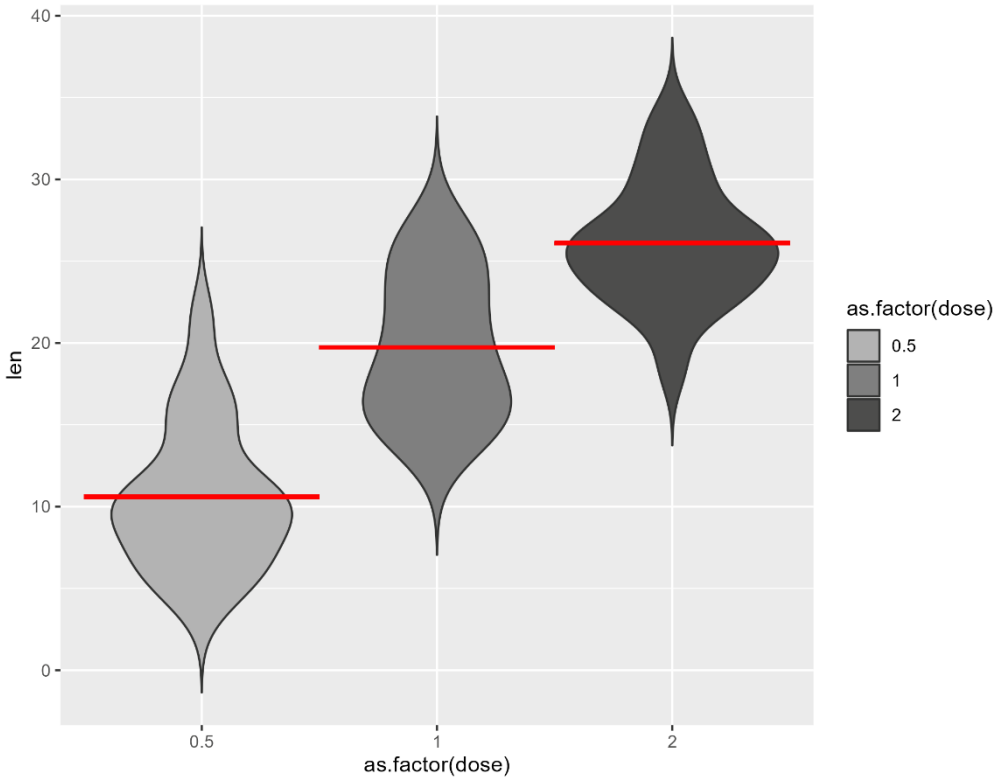


Figura 19.34. Gráfico violino com médias.

19.2.12 Barras de erro

Quando trabalhamos com dados numéricos e desejamos expor graficamente as médias, entende-se que os dados estão na normal. É interessante adicionar no gráfico um conjunto de linhas ou barras horizontais que identifique a amplitude de confiabilidade dessas médias, podendo ser através do desvio padrão, intervalo de confiança da média ou o erro padrão da média. Quando não estão na distribuição normal, é mais interessante montar um *boxplot*, onde estarão representados esse intervalo através dos quartis que rondam a mediana.

O erro padrão representa a variabilidade de uma média amostral em comparação com a média da população, servindo como uma métrica que avalia a confiabilidade da média amostral calculada. Para calcular o erro padrão, é necessário dividir o desvio padrão pela raiz quadrada do tamanho da amostra. Já o intervalo de confiança de uma média pode ser obtido através de um teste t ou pelos cálculos da ANOVA (verificar se há diferença entre médias) e o seu teste *post hoc* (para verificar onde há diferença entre médias). Já o desvio padrão, já sabemos a essa altura que usando

a função sd() conseguimos obtê-lo. Você escolhe o que você achar melhor, contudo, o uso do desvio padrão é mais frequente entre os pesquisadores.

Independente da variação que você desejar, o mais correto é calcular antes esses intervalos e deixá-los salvos em um objeto para compor o gráfico de barras, nas barras de erro.

Em nosso exemplo, retornaremos ao banco ToothGrowth. Iremos montar um *data frame* com a sequência das variáveis categóricas (supp e dose), juntamente com o cálculo das médias por grupos e das variações obtidas pelo erro padrão e pelo desvio padrão (DP), todas calculadas usando a função aggregate(), que subdivide o banco original para calcular separadamente grupos presentes em uma ou mais variáveis categóricas.

```
#Calculando as médias para cada conjunto: dose e supp
medias <- aggregate(len ~ dose + supp, data = ToothGrowth, FUN = mean)
medias$len #essas são as médias

#Calculando o DP
desvioPadrao <- aggregate(len ~ dose + supp, data = ToothGrowth, FUN = sd)
desvioPadrao$len #esses são os DP

#Função para o cálculo do erro padrão da média
erro_padrao_media <- function(dados){
  sd(dados) / sqrt(length(dados))
}

#Calculando o erro padrão de cada média
erroPadrao <- aggregate(len ~ dose + supp, data = ToothGrowth, FUN = erro_padrao_me-
dia)
erroPadrao$len #esses são os erros padrão das médias

#Montando data frame:
tabelaDeDados <- data.frame(
  dose = as.factor(rep(c(0.5, 1, 2.0),2)),
  supp = c(rep('OJ',3),rep('VC', 3)),
  medias = medias$len,
  DP_menos = medias$len - desvioPadrao$len,
  DP_mais = medias$len + desvioPadrao$len,
  erro_menos = medias$len - erroPadrao$len,
  erro_mais = medias$len + erroPadrao$len
)
```

Perceba que na montagem do *data frame* calculamos os intervalos somando e diminuindo DP e erro padrão de cada média.

Primeiramente, vamos usar a função geom_errorbar() para preencher os intervalos com as linhas de erro. Como temos grupos diferentes, vamos separar as barras utilizando o argumento "dodge" no parâmetro "position", na função geom_col() e na função geom_errorbar(). As linhas de erro são delimitadas quanto ao seu valor menor (a subtração) e seu valor maior (a soma). No

gráfico, a média (limite da barra criada por geom_col()) obviamente precisa estar no meio desse intervalo, caso contrário há algum erro nos cálculos.

Vamos trabalhar primeiro com desvio padrão. Vejamos:

```
#Figura 19.35
tabelaDeDados %>%
  ggplot(aes(x = supp, y = medias, fill = dose)) +
  geom_col(position = "dodge") +
  geom_errorbar(aes(ymin = DP_menos, ymax = DP_mais),
                position = position_dodge(),
                linewidth = 0.5) +
  scale_fill_manual(values = c('grey80', 'grey50','grey20'))
```

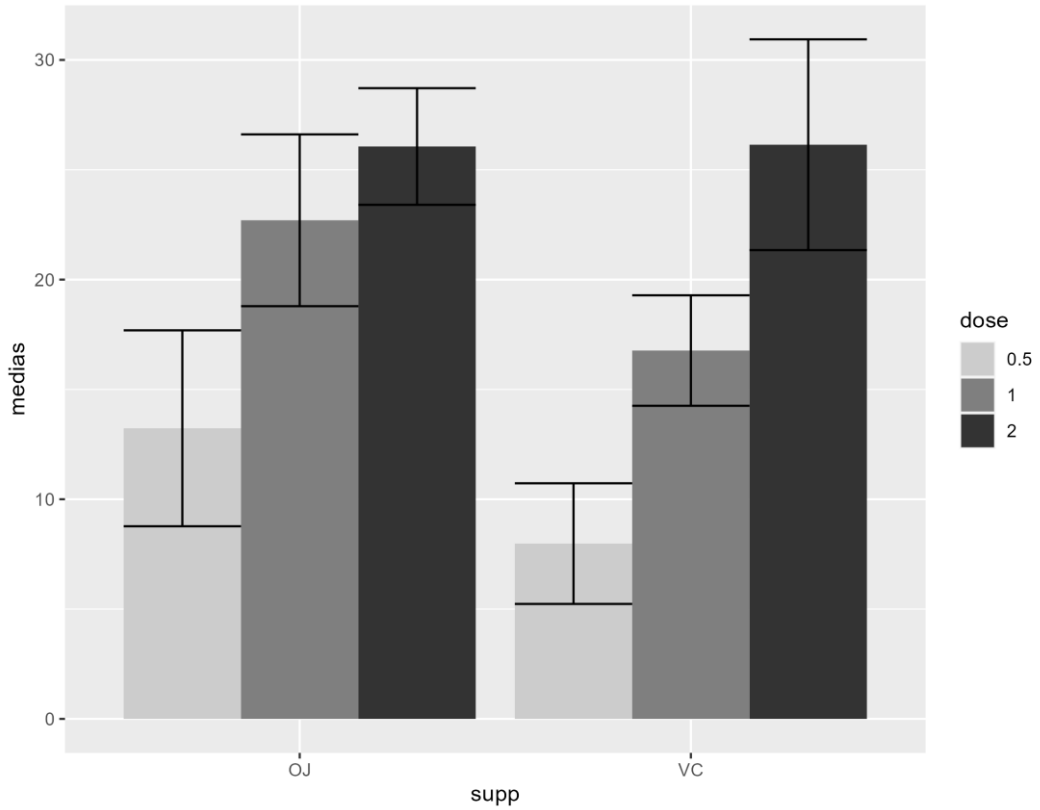


Figura 19.35. Gráfico de barras com barras de erro.

Vamos ver como ficará o gráfico se usarmos o erro padrão de cada média. Vamos pôr de azul para modificar um pouco:

```
#Figura 19.36
tabelaDeDados %>%
  ggplot(aes(x = supp, y = medias, fill = dose)) +
  geom_col(position = "dodge") +
  geom_errorbar(aes(ymin = erro_menos, ymax = erro_mais),
                position = position_dodge(),
                linewidth = 1,
                color = 'blue') +
  scale_fill_manual(values = c('grey80', 'grey50','grey20'))
```

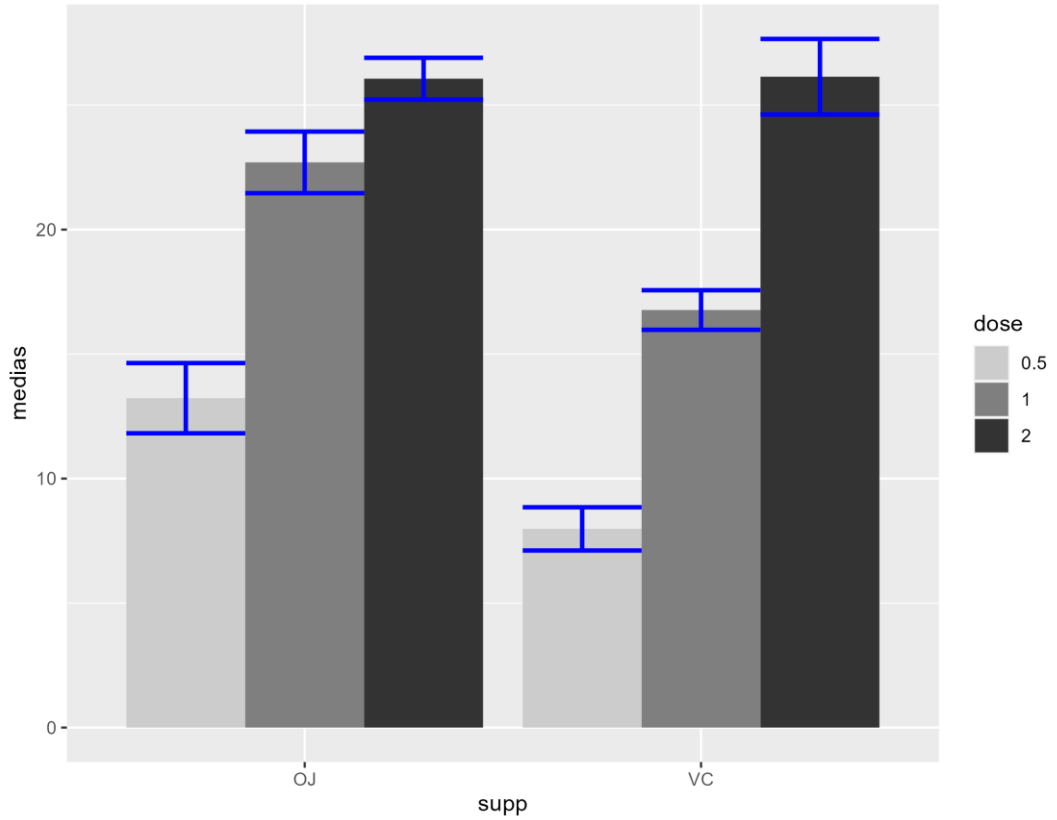


Figura 19.36. Gráfico de barras com barras de erro esteticamente modificadas.

Perceba que para esse exemplo de dados o erro padrão apresentou um intervalo menor, mostrando linhas para próximas entre as médias.

As mesmas barras/linhas de erro podem ser feitas utilizando geom_cross() – aparecerá como uma caixa semelhante à caixa do *boxplot* - ou geom_pointrange() – aparecerá como linhas verticais e o ponto central representando a média. Não são comuns seus usos, mas teste e verifique como ficará. Utilize 0.9 para o parâmetro "width", para que as linhas fiquem exatamente no meio das barras.

```
#Figura 19.37
tabelaDeDados %>%
  ggplot(aes(x = supp, y = medias, fill = dose)) +
  geom_col(position = "dodge") +
  geom_crossbar(aes(ymin = DP_menos, ymax = DP_mais),
                position = position_dodge(width = 0.9),
                alpha = 0, color = 'green4') +
  scale_fill_manual(values = c('grey80', 'grey50','grey20'))
```

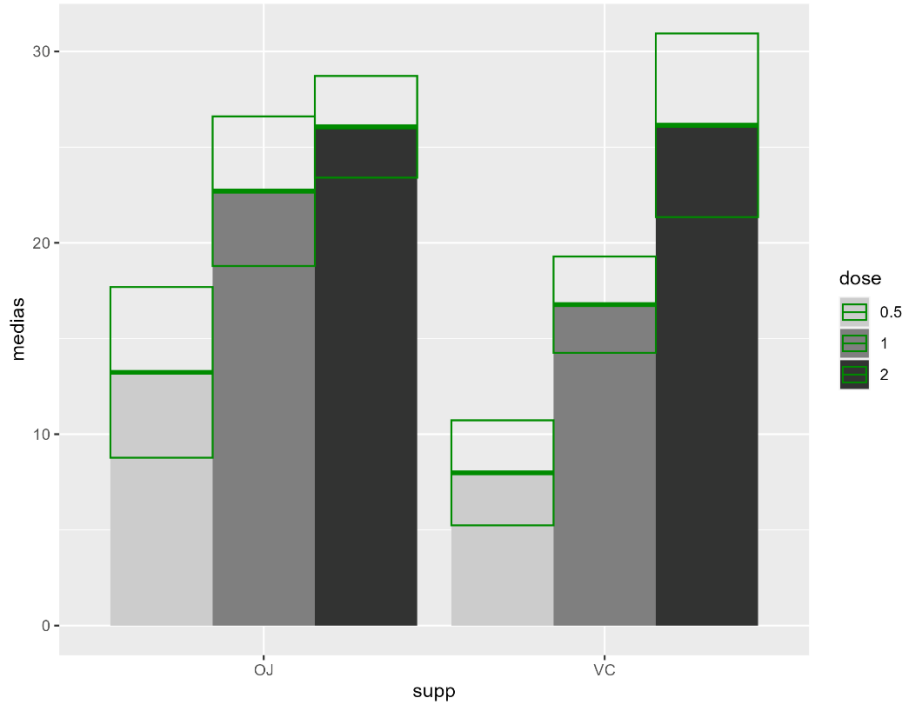


Figura 19.37. Gráfico de barras com caixas de erro.

```
#Figura 19.38
tabelaDeDados %>%
  ggplot(aes(x = supp, y = medias, fill = dose)) +
  geom_col(position = "dodge") +
  geom_pointrange(aes(ymin = DP_menos, ymax = DP_mais),
                  position = position_dodge(width = 0.9)) +
  scale_fill_manual(values = c('grey80', 'grey50','grey20'))
```

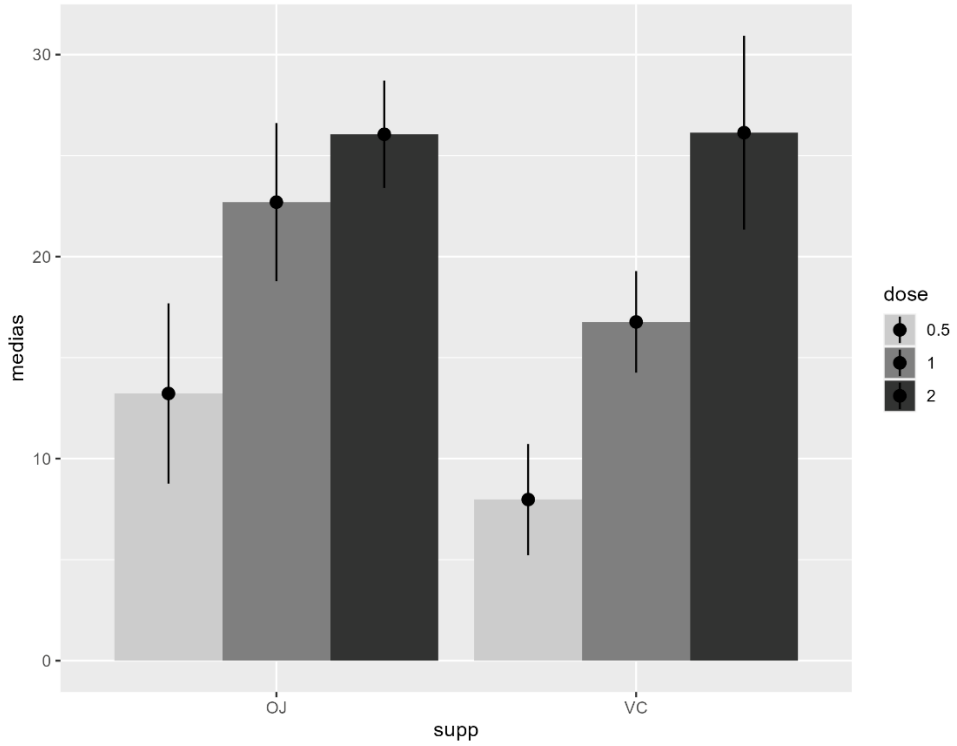


Figura 19.38. Gráfico de barras com ponto e linhas de erro.

19.2.13 Adição de textos nos gráficos

É de grande valia a adição de texto dentro dos gráficos. É possível pôr uma informação valiosa que seja do interesse do interpretador, como valores de médias ou medianas, o resultado do p-valor em comparações de médias, correlações e outras várias situações.

Podemos adicionar qualquer texto usando geom_text() ou geom_label().

Vamos continuar usando os dados que calculamos em tabelaDeDados logo acima. Usando geom_text(), vamos adicionar no gráfico o valor de cada média, que já está contido nesse *data frame*, na coluna chamada "medias".

```
#Figura 19.39
tabelaDeDados %>%
  ggplot(aes(x = supp, y = medias, fill = dose)) +
  geom_col(position = "dodge") +
  geom_errorbar(aes(ymin = DP_menos, ymax = DP_mais),
                position = position_dodge(),
                linewidth = 0.5) +
  geom_text(aes(label = medias),
    position = position_dodge(0.9),
    vjust = -1) +
  scale_fill_manual(values = c('grey80', 'grey50','grey20'))
```

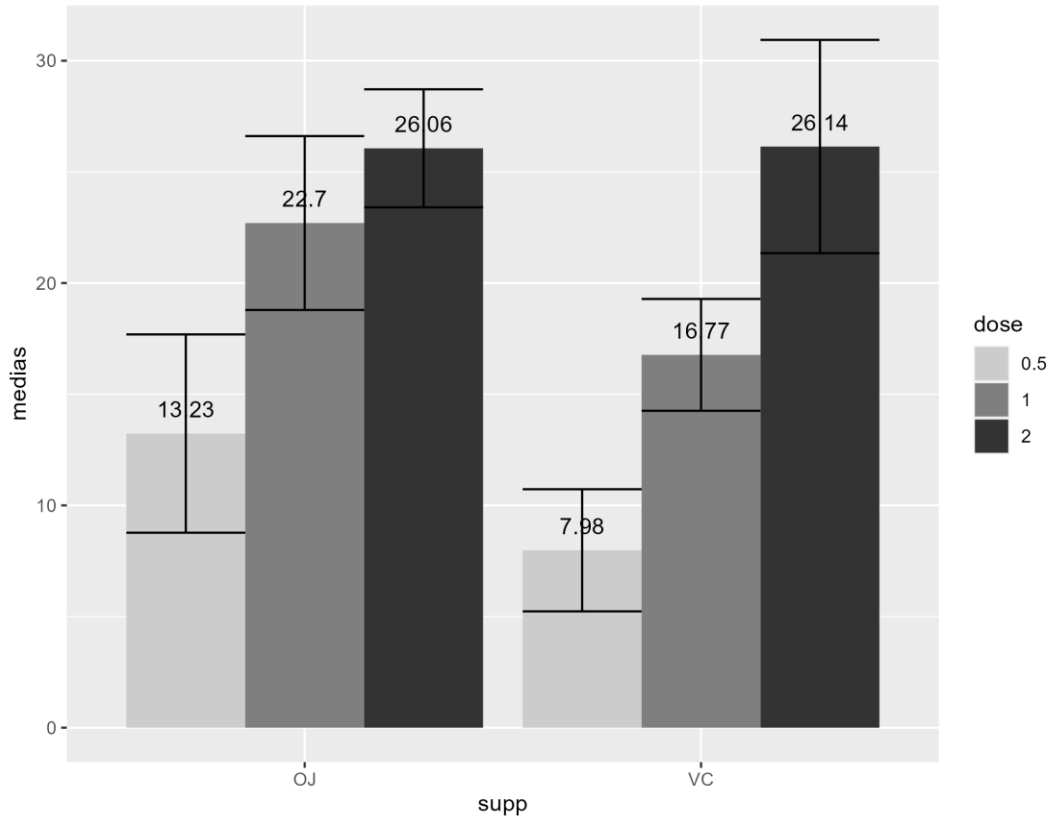


Figura 19.39. Gráfico de barras com barras de erro e valores de média.

A diferença entre geom_text() e geom_label() é que nessa última os valores aparecerão em uma caixa de texto. Também, geom_label() não responde ao parâmetro de angulação como geom_text(). Veja:

```
#Figura 19.40
tabelaDeDados %>%
  ggplot(aes(x = supp, y = medias, fill = dose)) +
  geom_col(position = "dodge") +
  geom_errorbar(aes(ymin = DP_menos, ymax = DP_mais),
                position = position_dodge(),
                linewidth = 0.5) +
  geom_label(aes(label = medias),
             position = position_dodge(0.9),
             vjust = -1, alpha = 0.5) +
  scale_fill_manual(values = c('grey80', 'grey50','grey20'))
```

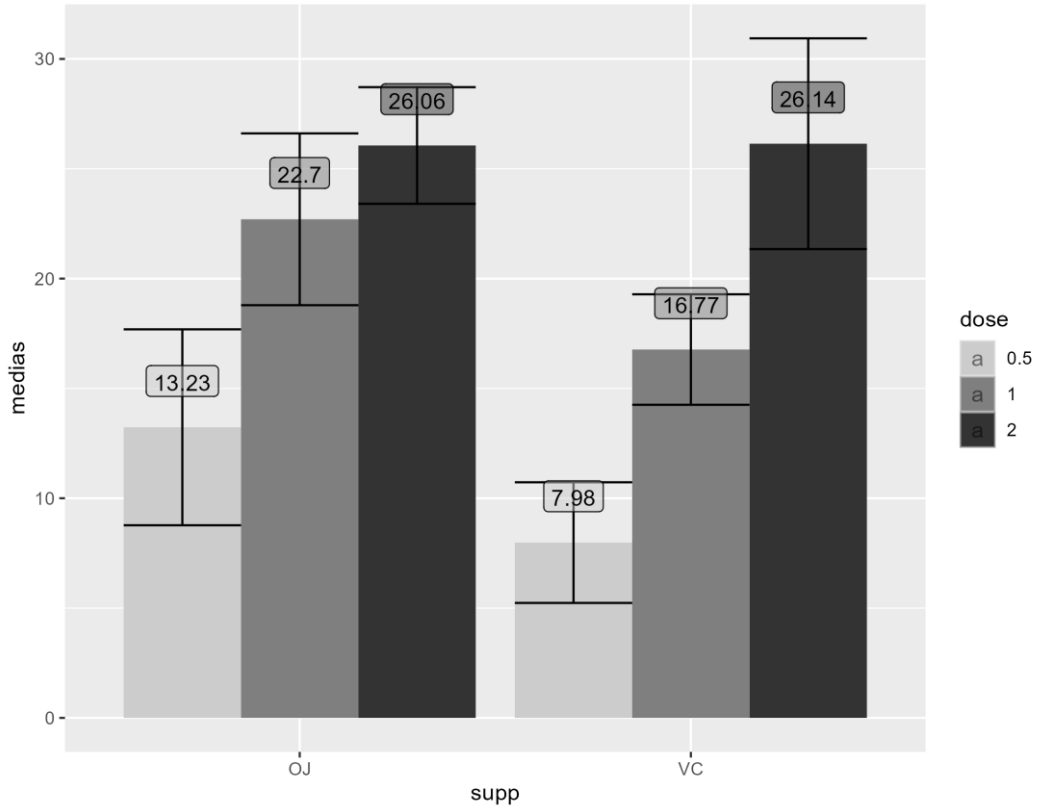


Figura 19.40. Gráfico de barras com barras de erro e valores de média em forma de rótulo.

Perceba que para especificar onde estavam as médias, tivemos que preencher o parâmetro "label" na estética das funções geom_text() e geom_label().

Outros parâmetros interessantes para serem utilizados:

- x = especificar onde o texto ficará em relação ao eixo x.
- y = especificar onde o texto ficará em relação ao eixo y.

- check_overlap = booleano (TRUE/FALSE), para ajustar os textos para que não fiquem sobrepostos.
- angle = numérico, cuja função serve para dar angulação ao texto.
- family = caractere; muda a fonte do texto. Especificar nome da fonte.
- size = numérico; tamanho da fonte.
- vajust = numérico; ajusta o texto mais para cima ou mais para baixo. Aceita valores negativos.
- fontface = caractere; podemos alterar o texto para que fique em itálico ou negrito.

Tipos de fontface:

- "plain" = normal.
- "bold" = negrito.
- "italic" = intálico.
- "bold.italic" = negrito e itálico

Vamos usar alguns desses parâmetros para exibir o valor de P do teste da ANOVA entre as espécies de íris quanto ao comprimento das sépalas. Usaremos, portanto, o banco iris. Vamos primeiramente calcular a ANOVA, expor seus resultados e calcular individualmente a comparação dos pares de médias (testes post hoc) através do teste Tukey.

```
#modelo da ANOVA
anova_result <- aov(Sepal.Length ~ Species, data = iris)

# Exibindo os resultados da ANOVA
summary(anova_result)
             Df Sum Sq Mean Sq F value Pr(>F)
Species       2  63.21  31.606   119.3 <2e-16 ***
Residuals   147  38.96   0.265
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

#teste Tukey (um dos testes post hoc)
TukeyHSD(anova_result)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Sepal.Length ~ Species, data = iris)

$Species
                     diff       lwr       upr p adj
versicolor-setosa    0.930 0.6862273 1.1737727     0
virginica-setosa     1.582 1.3382273 1.8257727     0
virginica-versicolor 0.652 0.4082273 0.8957727     0
```

Primeiramente, a ANOVA indica que há médias diferentes, pois retornou um p-valor < 0,001, em Pr(>F) sendo <2e-16 ***. Para as comparações par a par, todos os valores de p ajustados também foram significativos, em p adj com valores 0 (significa p-valor < 0,001). Em suma, todas as médias de cada espécie são diferentes entre si.

Vamos grosseiramente pôr esses resultados de P no gráfico, para que percebamos possibilidade de configuração de texto. Repito: grosseiramente.

```
#Figura 19.41
iris %>%
  ggplot(aes(x = Species,y = Sepal.Length)) +
  geom_boxplot() +
  geom_text(aes(x = 2, y = 8, label = 'ANOVA p<0.001'), size = 7) +
  geom_text(aes(x = 2, y = 7.5, label = 'setosa-virginica p<0.001'),
            angle = 45) +
  geom_text(aes(x = 1.5, y = 6, label = 'setosa-versicolor p<0.001'),
            angle = -45, family = 'serif') +
  geom_text(aes(x = 2.5, y = 6, label = 'versicolor-virginica p<0.001'),
           angle = 90, family = 'mono')
```

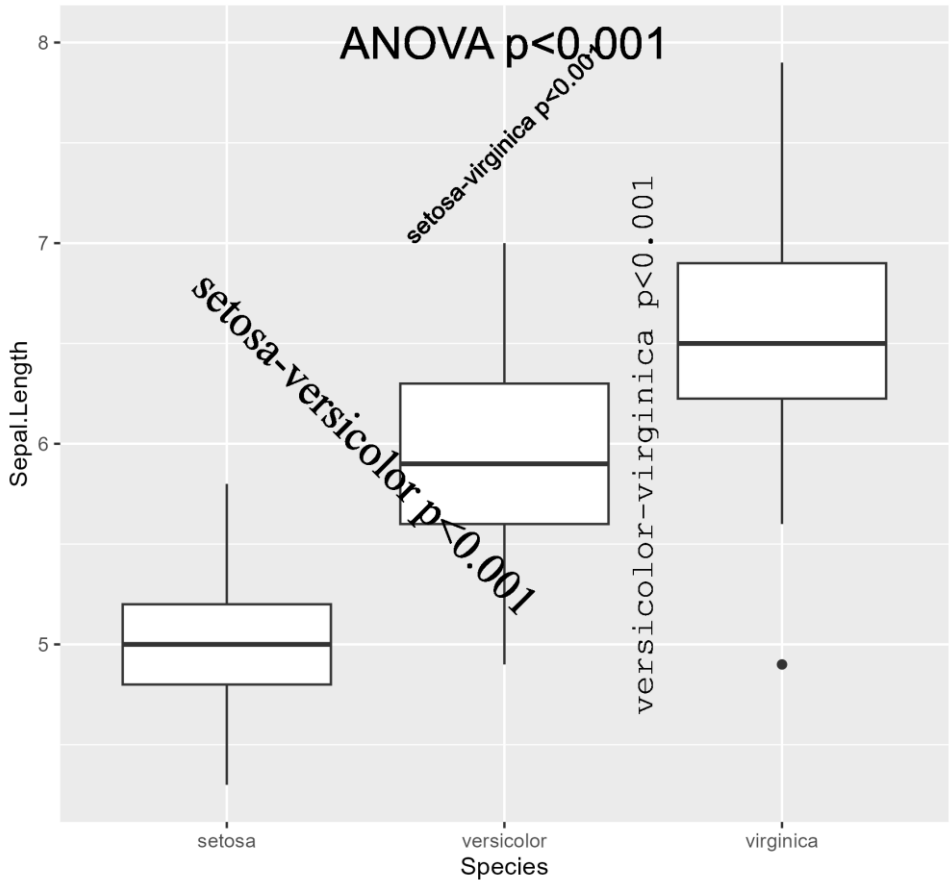


Figura 19.41. Gráfico com textos soltos em diversas possibilidades.

Se o R estiver com problemas de reconhecer o nome da fonte que você esteja digitando, use a função windowsFonts() para verificar quais as fontes que o seu R está reconhecendo.

```
windowsFonts()

$serif
[1] "TT Times New Roman"

$sans
[1] "TT Arial"

$mono
[1] "TT Courier New"
```

Nesse momento só três fontes o meu R consegue identificar. Para Times New Roman preciso pôr como argumento "serif"; a Arial como "sans" (normalmente já vem como padrão); e, Courier New devo pôr "mono". Caso você precise de uma fonte específica, use o caminho da pasta e o nome da fonte para especificá-la. Lembrando que fontes normalmente vêm com extensões do tipo .ttf ou .otf.

19.2.14 Segmentos de reta

Seguimentos de reta podem ser muito úteis na construção de um gráfico didático e autoexplicativo. Há quatro funções que podemos usar para adicionar no gráfico linhas, seja elas verticais, horizontais ou inclinadas.

A função geom_abline() tem como objetivo adicionar uma reta que é a representação gráfica de uma equação do 1° grau do tipo $y = ax + b$, onde a é a angulação da reta (coeficiente angular da reta) e b o intercepto (ou termo independente, o qual a reta passará no eixo y). Para isso, precisamos estipular valores numéricos para os parâmetros "slope" (do inglês, inclinação) e "intercept", respectivos aos coeficientes.

A função geom_hline() acrescenta uma linha horizontal, bastando estipular valor para o parâmetro "yintercept". Já a função geom_vline() acrescenta uma linha vertical, bastando estipular o valor desejado para o parâmetro "xintercept".

A função geom_segment() pode ser mais específica do que as demais, pois a ela necessita estabelecer onde começar e onde terminar a reta. Para isso precisamos estipular valores para os parâmetros "x", "xend", "y" e "yend".

Há uma diferença importante entre elas: geom_abline(), geom_hline() e geom_vline() aplicarão segmento de reta contínuo, ou seja, que cortará todo o gráfico. Já geom_segment() foi projetada para ter uma reta rigorosamente bem definida, tendo seu início e fim de reta.

Veja o exemplo a seguir:

```
#Figura 19.42
#seguimentos de reta em um só gráfico
ggplot() +
  geom_abline(intercept = 7, slope = -1,
              color = 'red', linewidth = 4) +
  geom_hline(yintercept = 5, color = 'blue', linewidth = 4) +
  geom_vline(xintercept = 3, color = 'green', linewidth = 4) +
  geom_segment(aes(x = 2, xend = 7, y = 2, yend = 7),
               color = 'chocolate', linewidth = 4) +
  scale_x_continuous(limits = c(0,10)) +
  scale_y_continuous(limits = c(0,10))
```

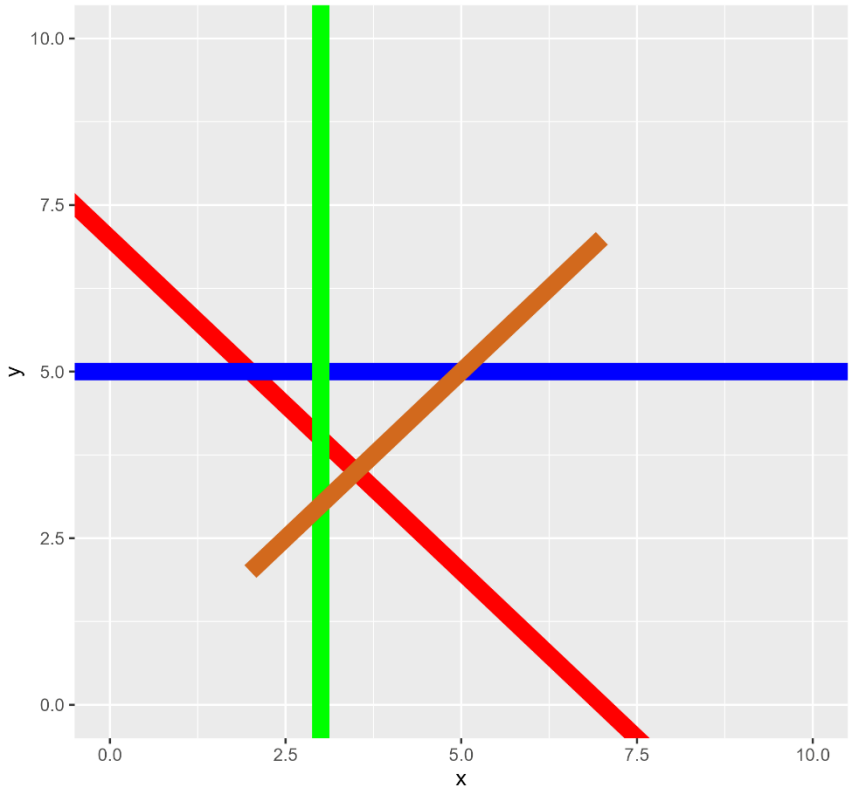


Figura 19.42. Vários segmentos de reta.

Sobre as funções scale_ iremos falar mais adiante, mas basicamente estão servindo para mudar as escalas dos eixos. Vamos nos ater no momento apenas nas retas. A primeira coisa a se observar é que realmente todas as funções geraram retas que ligam as extremidades do gráfico, com exceção de geom_segment(), que delimitou a reta em um comprimento menor. Outra coisa interessante é que com essa imagem podemos enxergar a construção gráfica em camadas. Perceba que as linhas estão sobrepostas de forma que segue a sequência de entrada das funções do código, ou seja, no gráfico aparece a reta vermelha que foi escrita usando geom_abline(), que aparece em

primeiro no código. Já a que está mais por cima, a de cor chocolate, foi escrita por último com geom_segment().

Então, veja que podemos sobrepor gráficos diferentes, sendo que o último ficará visualmente na frente dos demais. Em tais sobreposições é muito importante ficarmos atentos à escala, pois gráficos podem não aparecer por estarem fora da escala dos eixos x ou y.

Sobre os parâmetros interessantes que podemos alterar, são os mesmos dos gráficos de linhas, como "linetype", "linewidth", "color" e "alpha".

Por fim, vamos dar um exemplo mais concreto do uso de segmentos de reta para a bioestatística. Já vimos que usando o banco iris, comparando estatisticamente o comprimento das sépalas entre as três espécies do banco são todas médias estatisticamente diferentes entre si. Vamos pôr essas informações em um gráfico do tipo *jittered points*, colocar linhas vermelhas horizontais demonstrando ao leitor onde estão as médias de cada espécie, e colocar retas horizontais entre as espécies para estabelecer as comparações de médias seguida dos seus p-valores respectivos. Veja como ficará:

```
#Figura 19.43
iris %>%
  ggplot(aes(x = Species,y = Sepal.Length)) +
  geom_jitter(width = 0.2) +
  geom_segment(aes(x = 0.5, xend = 1.5,
                   y = 5.006, yend = 5.006),
               color = 'red', linewidth = 1) +
  geom_segment(aes(x = 1.5, xend = 2.5,
                   y = 5.936, yend = 5.936),
               color = 'red', linewidth = 1) +
  geom_segment(aes(x = 2.5, xend = 3.5,
                   y = 6.588, yend = 6.588),
               color = 'red', linewidth = 1) +

  #adiconando p-valores
  geom_segment(aes(x = 1, xend = 2, y = 7, yend = 7)) +
  geom_segment(aes(x = 2, xend = 3, y = 7.5, yend = 7.5)) +
  geom_segment(aes(x = 1, xend = 3, y = 8, yend = 8)) +
  geom_text(aes(x = 1.5, y = 7.1), label = 'p<0.001') +
  geom_text(aes(x = 2.5, y = 7.6), label = 'p<0.001') +
  geom_text(aes(x = 2, y = 8.1), label = 'p<0.001')
```

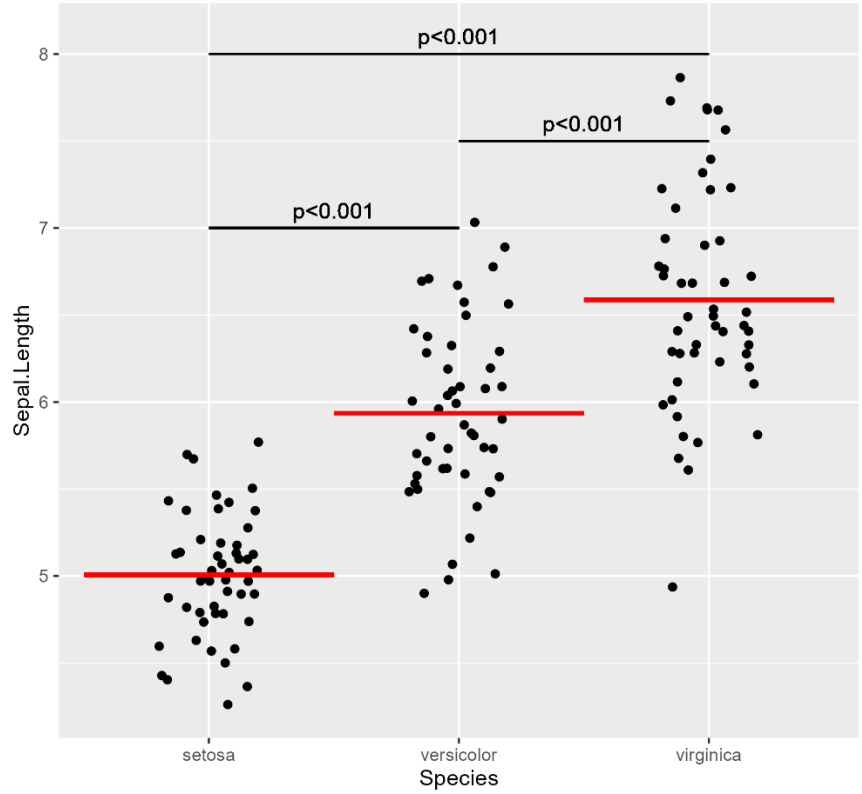


Figura 19.43. *Jitter plot* com p-valores das comparações de médias.

19.2.15 Mudando a escala dos eixos gráficos

As escalas são as marcações numéricas ou classificatórias que apresentam os eixos. Se a variável for entendida como numérica, o eixo terá intervalos contínuos ou discretos. Quando uma variável é entendida como categórica ou fator, a escala dela no eixo será também categoria quantificada pelo número de classes presentes na variável.

As funções do ggplot2 que remetem às modificações de escala começam com `scale_`. Se quisermos mudar a escala do eixo x, então ele apresentará mais um caractere para especificar que é a escala do eixo x: `scale_x_`. Depois disso estará descrito o tipo da escala a ser mudada, se for por exemplo valores contínuos, teremos a função final `scale_x_continuos()`. Os sufixos mais comuns são `_manual()`, para mudar a escala manualmente, `_continuos()` que você já sabe sua finalidade, `_discrete()` que muda a escala da variável quando discreta e `_date()` que muda a sequência de data em uma série temporal.

É possível mudar escalas de tipos diversos. Veja outros prefixos e ao que ela está associada:

- `scale_y_`: muda a escala do eixo y;
- `scale_color_`: muda a escala da variável designada para mudança de cores;

- scale_fill_: muda a escala da variável de preenchimento;
- scale_alpha_: muda a escala da variável de transparência;
- scale_linetype_: muda a escala da variável designada para alterar tipos de linhas diferentes;
- scale_linewidth_: muda a escala da variável de espessura de linha;
- scale_radius: muda a escala de área (geralmente de pontos);
- scale_shape_: muda a escala de formas diferentes;
- scale_size_: muda a escala de tamanho.

Vamos entender melhor na prática. Voltemos ao gráfico sobre comprimento dental, o ToothGrowth. Abaixo, o gráfico está "seco", com escalas automáticas do próprio ggplot2.

```
#Figura 19.44
ToothGrowth %>%
  ggplot(aes(x=as.factor(dose), y=len, fill = as.factor(dose))) +
  geom_violin()
```

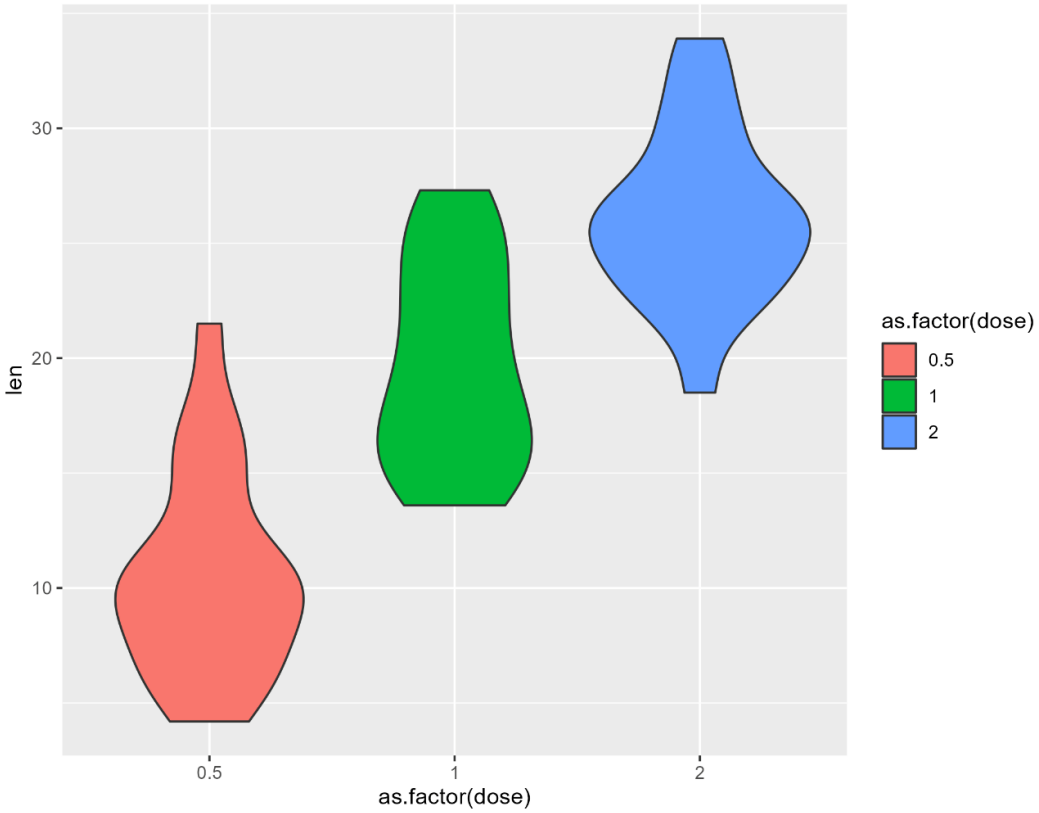


Figura 19.44. Gráficos violino padrão, por grupos.

Primeiramente, vamos pensar sobre o eixo x. Vemos que a escala está como fator (categorias), e não como número verdadeiramente (se ater a isso é importante). Temos então três classes: a dose de 0,5 mL/dia, a dose de 1 mL/dia e a dose de 2 mL/dia, mas estão descritas originalmente

como: 0.5, 1 e 2. Vamos mudar isso. Dá para melhorar a apresentação dessa escala usando scale_x_discrete(), e entenda porque aqui está como discreta: o R entende como valores discretos e até ordinais. Vamos modificar pondo a unidade de medida das doses. Veja:

```
#Figura 19.45
ToothGrowth %>%
  ggplot(aes(x=as.factor(dose), y=len, fill = as.factor(dose))) +
  geom_violin() +
  scale_x_discrete(labels = c('0.5' = 'Dose 0,5 mL/dia',
                              '1' = 'Dose 1,0 mL/dia',
                              '2' = 'Dose 2,0 mL/dia'))
```

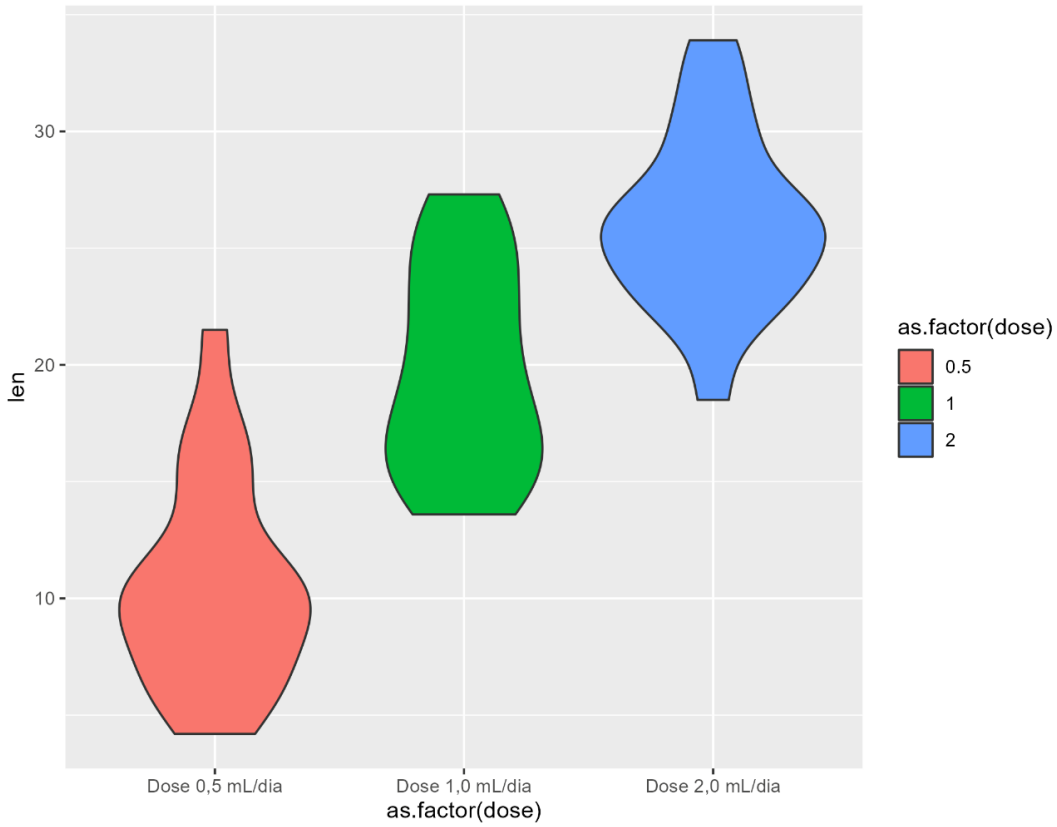


Figura 19.45. Gráficos violino com mudança de escala no eixo x.

Perceba que a escala segue exatamente a ordem das dosagens como escritas no código.

Agora, vamos pensar sobre o eixo y. Nele há uma variável numérica do tipo contínua. Vemos que os intervalos vão de 10 em 10 (por isso vemos a sequência 10, 20, 30). Dá para estimar, mas não dá para saber exatamente o número que começa e que termina essa escala. Como se trata de variável numérica, vamos modificar a escala usando scale_y_continuos(). Vamos alterar a escala para que o gráfico tenha no eixo y um intervalo entre 0 e 50, e que mude de 5 em 5. Para isso, devemos mexer em dois parâmetros: "breaks" e "limits". Em "breaks" estamos especificando

justamente os intervalos e "limits" estaremos especificando os limites desse intervalo (de 0 a 50). Por vezes usar só "breaks" é suficiente, mas terá momentos que os limites desejados não aparecerão no gráfico e será necessário especificar "limits" obrigatoriamente. Veja:

```
#Figura 19.46
ToothGrowth %>%
  ggplot(aes(x=as.factor(dose), y=len, fill = as.factor(dose))) +
  geom_violin() +
  scale_x_discrete(labels = c('0.5' = 'Dose 0,5 mL/dia',
                              '1' = 'Dose 1,0 mL/dia',
                              '2' = 'Dose 2,0 mL/dia')) +
  scale_y_continuous(breaks = seq(0,50,5), limits = c(0,50))
```

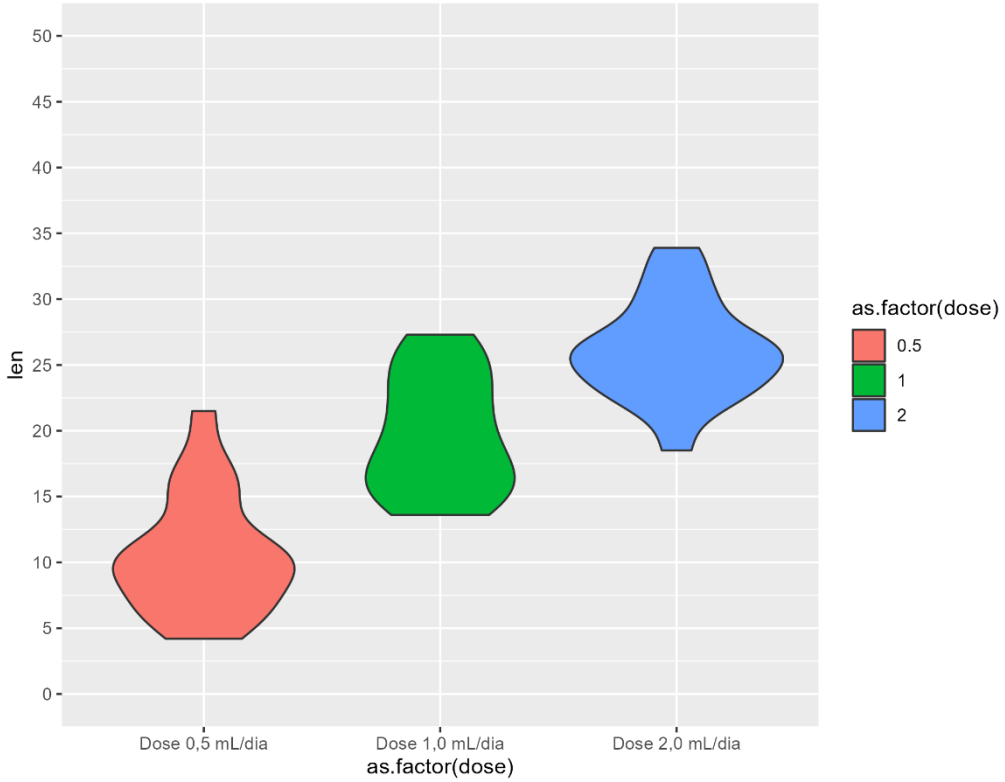


Figura 19.46. Gráficos violino com mudança de escala no eixo x e y.

Nosso gráfico está ficando cada vez mais compreensível a quem está analisando. Por último, vamos mudar a escala de preenchimento usando scale_fill_manual(), porque quero lhes mostrar que aqui é possível mudar padrões de cores e também o rótulo da legenda diretamente com a escala de preenchimento manual. Modificaremos os parâmetros "values" (valores) e "labels" (rótulos). Veja:

```
#Figura 19.47
ToothGrowth %>%
  ggplot(aes(x=as.factor(dose), y=len, fill = as.factor(dose))) +
```

```
  geom_violin() +
  scale_x_discrete(labels = c('0.5' = 'Dose 0,5 mL/dia',
                              '1' = 'Dose 1,0 mL/dia',
                              '2' = 'Dose 2,0 mL/dia')) +
  scale_y_continuous(breaks = seq(0,50,5), limits = c(0,50)) +
  scale_fill_manual(values = c('gray70','gray50','gray30'),
                    labels = c('0.5' = 'Dose fraca',
                               '1' = 'Dose intermediária',
                               '2' = 'Dose alta'))
```

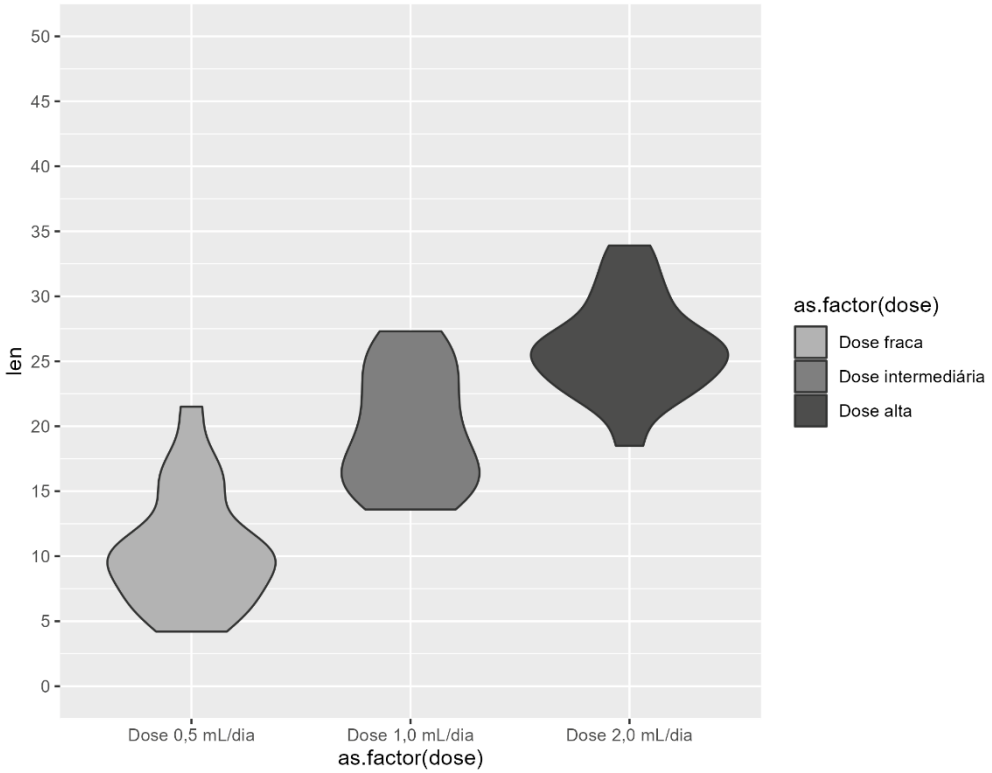


Figura 19.47. Gráficos violino com mudança de escala no eixo x e y e de preenchimento.

19.2.16 Alterando os títulos dos eixos e legenda

Muito comum é precisarmos mudar os títulos, seja o título principal, o título dos eixos ou mesmo o título da legenda. Para isso, basta usarmos a função labs() e nela mudarmos os parâmetros referentes a cada um deles. Para mudar o título do gráfico basta especificar em "title", para o título do eixo x basta especificar em "x", para o título do eixo y basta modificar em "y", para mudar o título da legenda gerado por preenchimento basta alterar em "fill", se for por cor basta alterar em "color". Você ainda pode adicionar subtítulo especificando em "subtitle" ou pôr uma nota na parte inferior do gráfico especificando isso no parâmetro "caption". Veja as possibilidades:

```
#Figura 19.48
ToothGrowth %>%
  ggplot(aes(x=as.factor(dose), y=len, fill = as.factor(dose))) +
```

```
  geom_violin() +
  scale_x_discrete(labels = c('0.5' = 'Dose 0,5 mL/dia',
                              '1' = 'Dose 1,0 mL/dia',
                              '2' = 'Dose 2,0 mL/dia')) +
  scale_y_continuous(breaks = seq(0,50,5), limits = c(0,50)) +
  scale_fill_manual(values = c('gray70','gray50','gray30'),
                    labels = c('0.5' = 'Dose fraca',
                               '1' = 'Dose intermediária',
                               '2' = 'Dose alta')) +
  labs(title = "Gráficos violinos",
       x = 'Dosagens seriadas',
       y = 'Comprimento do dente',
       fill = 'Tipos de doses',
       subtitle = 'Será que subtítulos são realmente necessários?',
       caption = 'A nota ficou aqui, mas achei estranho.')
```

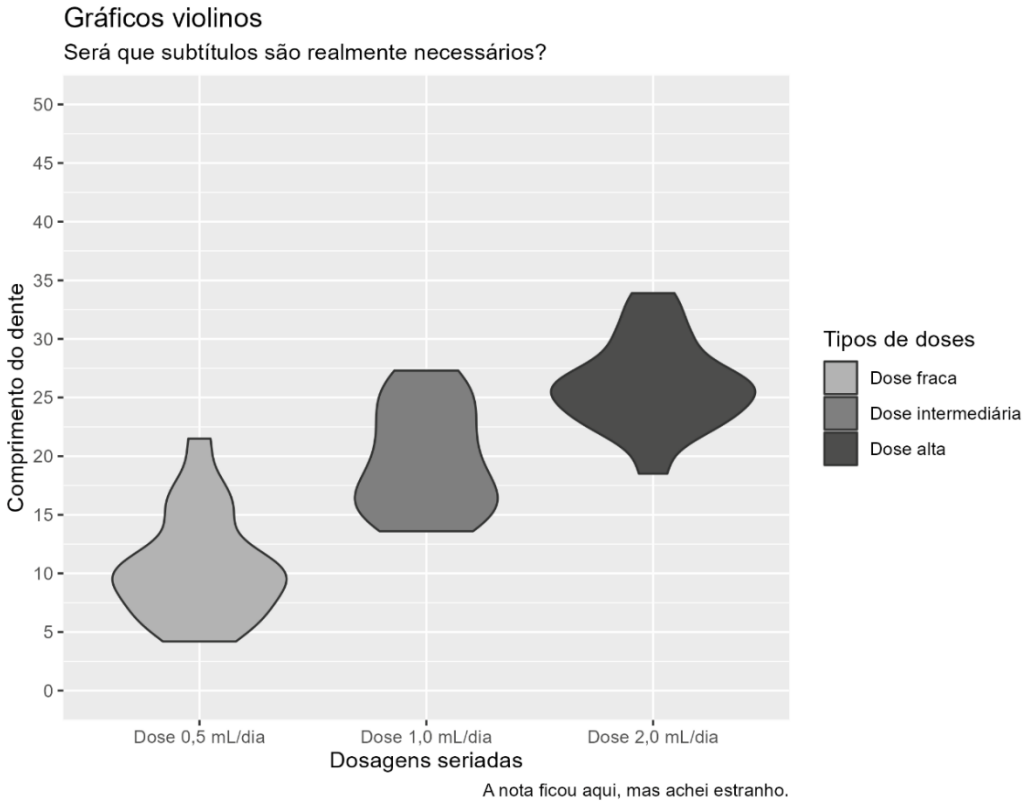


Figura 19.48. Gráficos violino com mudança de títulos, subtítulo, *caption*, título dos eixos e da legenda.

19.2.17 Alterando o tamanho da fonte dos títulos e das escalas

Os elementos textuais comuns de um gráfico são considerados como parte do tema. Então, para alterar o tamanho da fonte, cor, angulação e outras características do título do gráfico, títulos

dos eixos, título da legenda e o texto das escalas, devemos modificar isso dentro da função theme(). Nela há diversos parâmetros úteis para manipular características gráficas. Iremos os mais utilizados, mas há vários outros de fácil compreensão. Para ver todos eles e suas funcionalidades, busque na documentação da função theme(). Há muitas possibilidades por lá.

Parâmetros úteis para modificar características textuais em theme():

- plot.title = manipulação do texto do título do gráfico.
- axis.title.x = manipulação do texto do título do eixo x.
- axis.title.y = manipulação do texto do título do eixo y.
- legend.title = manipulação do texto do título da legenda.
- axis.text.x = manipulação do texto da escala do eixo x.
- axis.text.y = manipulação do texto da escala do eixo y.
- legend.text = manipulação dos textos da legenda.

Como queremos mexer no elemento textual desses títulos e escalas, precisamos adicionar a função acessória element_text() após o sinal de igualdade. Dentro dessa função podemos alterar diversas características, como a fonte do texto, o tamanho, a angulação, a cor, a forma, ajustar verticalmente ou horizontalmente o texto no gráfico. Os parâmetros para element_text() são:

- family = altera o tipo de fonte.
- size = altera o tamanho do texto.
- angle = altera a angulação do texto.
- color = altera a cor.
- face = altera a forma, se negrito ou itálico
- vjust = ajusta o texto verticalmente ("joga" o texto mais para cima ou para baixo).
- hjust = ajusta o texto horizontalmente ("joga" o texto mais para a esquerda ou para a direita).

Tome cuidado quando for ajustar o texto usando vjust ou hjust. Dependendo do valor que se coloca, o texto pode sair da parte visível do gráfico, o que pode parecer que houve *bug* e o texto não foi adicionado ao gráfico. Sugiro começar testando valores pequenos como 1 ou 2, ou -1 ou -2.

Veja o exemplo a seguir, com modificações exageradas para que você perceba o que e onde foi alterado.

```
#Figura 19.49
ToothGrowth %>%
  ggplot(aes(x=as.factor(dose), y=len, fill = as.factor(dose))) +
  geom_violin() +
  scale_x_discrete(labels = c('0.5' = 'Dose 0,5',
                              '1' = 'Dose 1,0',
                              '2' = 'Dose 2,0')) +
  scale_y_continuous(breaks = seq(0,50,5), limits = c(0,50)) +
  scale_fill_manual(values = c('gray70','gray50','gray30'),
                    labels = c('0.5' = 'Dose fraca',
                               '1' = 'Dose intermediária',
                               '2' = 'Dose alta')) +
  labs(title = "Gráficos violinos",
       x = 'Dosagens seriadas',
       y = 'Comprimento do dente',
       fill = 'Tipos de doses') +
  theme(
    #modificado letra dos títulos
    plot.title = element_text(color = 'blue'),
    axis.title.x = element_text(size = 14, vjust = 10),
    axis.title.y = element_text(size = 14, family = 'mono'),
    legend.title = element_text(size = 14, hjust = -5),
    #modificando letra das escalas
    axis.text.x = element_text(size = 12, angle = 90),
    axis.text.y = element_text(size = 12, face = 'italic'),
    legend.text = element_text(size = 12, color = 'red')
    )
```

Figura 19.49. Gráficos violino com fontes modificadas.

O gráfico está horrível, mas dá para entender perfeitamente o que os comandos fizeram. Primeiramente, em “plot.title” pedimos para mudar a cor do título do gráfico para azul. Em “axis.title.x” solicitamos que o tamanho da letra do título do eixo x fosse 14 e ajustasse-o verticalmente para cima do normal com valor 10. Em “axis.title.y” modificamos o tamanho do título do eixo y e trocamos a fonte para “Courier New”. Com “legend.title” ajustamos negativamente o título da legenda, o que empurra-o para a esquerda e acabou ficando dentro do gráfico (eu já falei que está horrível?). Com “axis.text.x” nos especificamos o tamanho de letra 12 e com angulação de 90° o texto da escala do eixo x. Com “axis.text.y” nós também deixamos em tamanho 12 e mudamos o tipo de letra para itálico. Por fim, com “legend.text” nós mudamos a cor das classes da legenda e deixamos no tamanho 12.

19.2.18 Cores

O uso de cores nos gráficos é elemento fundamental. O R já traz consigo algumas cores básicas para uso, as quais podemos escolhê-las apenas escrevendo seus nomes em inglês. Considero as mais úteis: "white" (branco), "black" (preto), "red" (vermelho), "green" (verde), "blue" (azul), "yellow" (amarelo), "orange" (laranja), "purple" (roxo) e "violet" (violeta). Além dessas, tem a paleta de cinza, que começa em "gray0" (o tom mais escuro), "gray1", "gray2", "gray3", "gray4", e assim sucessivamente até "gray100" (o tom de cinza mais claro).

Para ver todos os possíveis nomes das cores básicas do R, basta rodar a função colors().

```
colors()
[1]"white"          "aliceblue"      "antiquewhite"   "antiquewhite1"
[5]"antiquewhite2"  "antiquewhite3"  "antiquewhite4"  "aquamarine"
[9]"aquamarine1"    "aquamarine2"    "aquamarine3"    "aquamarine4"

(...)
```

Apesar de começar com *white*, todas as demais estão em ordem alfabética. Sabendo um pouco as cores em inglês, dá para ter uma noção de qual cor se trata cada uma delas. Outra forma interessante de saber melhor sobre essas cores básicas é que quando digitamos seus nomes ao longo do código, o RStudio no Windows (nas versões mais atuais) identifica e colore o seu nome com a sua cor respectiva no próprio código. Veja o exemplo de uso abaixo:

```
#Figura 19.50
iris %>%
  ggplot(aes(x = Petal.Length, y = Sepal.Length, color = Species)) +
  geom_point() +
  scale_color_manual(values = c('orange', 'blue', 'red'))
```

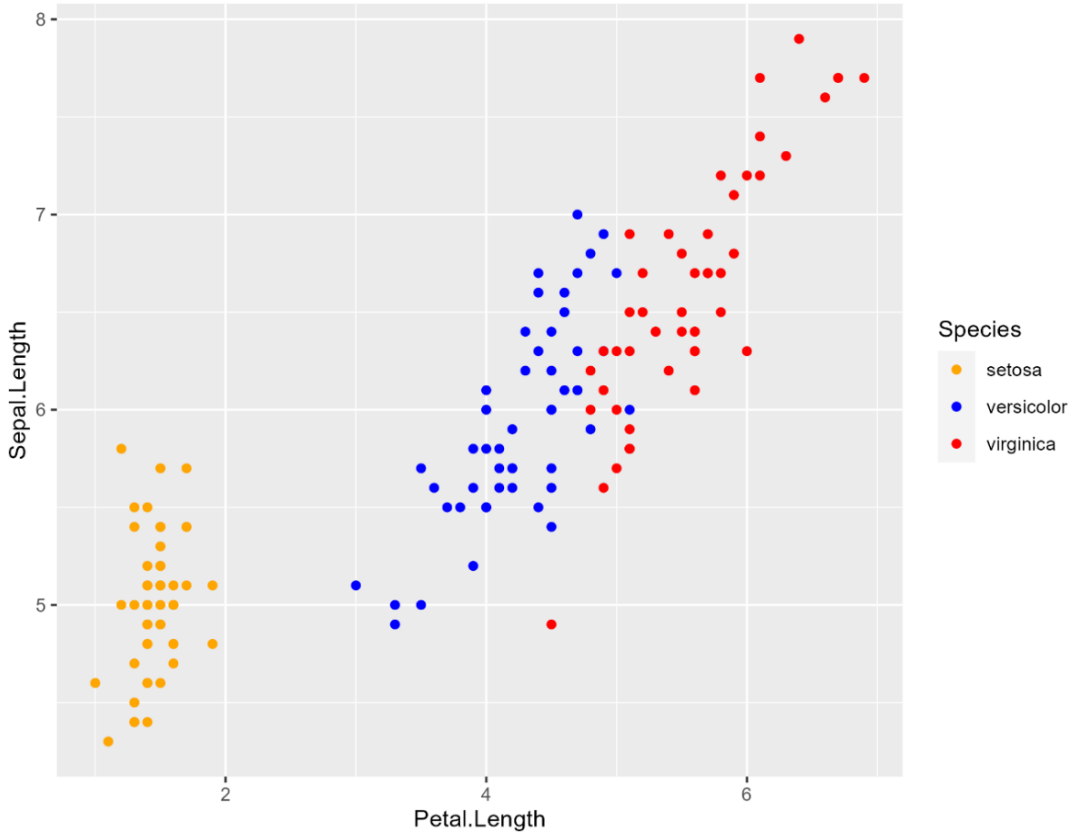


Figura 19.50. Gráficos de pontos com escala de cores modificadas manualmente com nomes.

Outro padrão muito interessante para cores é o sistema hexadecimal, muito utilizado em outras linguagens de programação. Como o nome sugere, cada cor consiste em uma combinação de seis caracteres alfanuméricos, representando a quantidade de vermelho, verde e azul (modelo RGB = *Red, Green, Blue*). Cada par de caracteres hexadecimais representa uma intensidade de cor, variando de 00 (sem intensidade) a FF (máxima intensidade). Por exemplo, o código hexadecimal #FF0000 representa vermelho puro, enquanto #00FF00 representa verde puro e #0000FF representa azul puro. Por fim, perceba que esse padrão sempre começa com # (cerquilha ou famigerado *hashtag*). Use o site Color-hex (www.color-hex.com) para escolher qualquer cor e obter seu código em hexadecimal.

Veja exemplo:

```
#Figura 19.51
iris %>%
  ggplot(aes(x = Petal.Length, y = Sepal.Length, color = Species)) +
  geom_point() +
  scale_color_manual(values = c('#6a329f', '#8fce00', '#bf9000'))
```

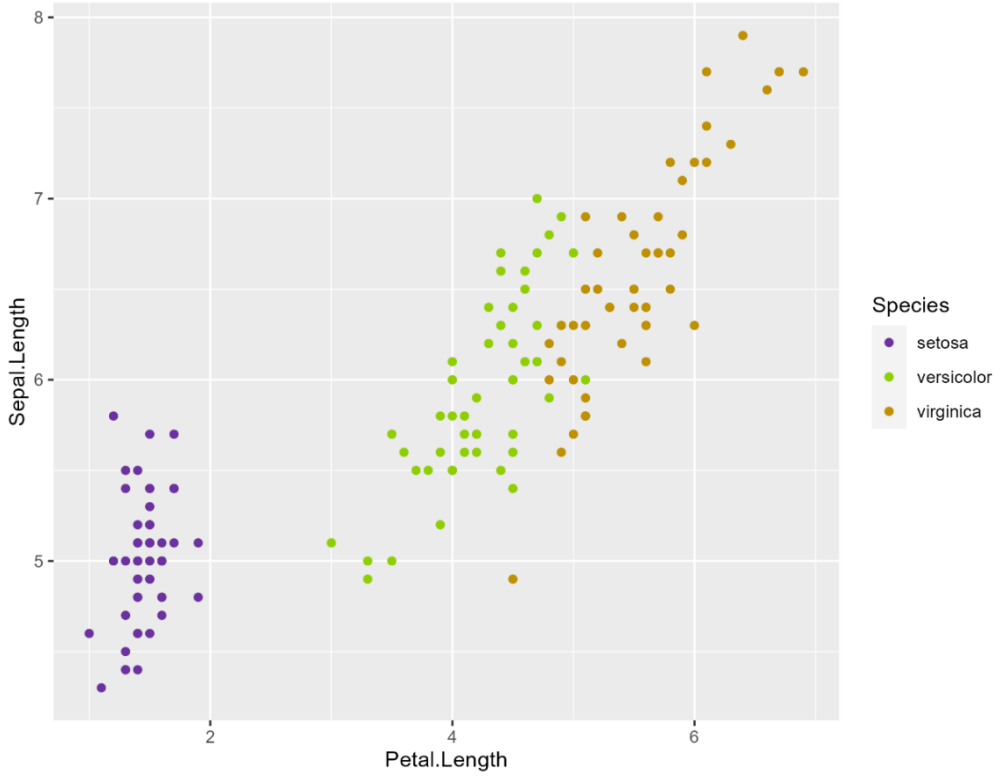


Figura 19.51. Gráficos de pontos com escala de cores modificadas manualmente em hexadecimal.

Há revistas científicas que exigem um padrão de cores em suas figuras. Descobrir suas paletas é importante. Quando não houver essa exigência, apele pela parcimônia visual. Figuras carregadas de cores pode ser mais poluição visual do que um atrativo didático. Dica: use primeiro a escala de cinza; caso fique de difícil compreensão, comece a usar cores básicas como o vermelho e o azul.

Por vezes estabelecer cor por cor se tornará uma tarefa chata e cansativa. Sem contar que estabelecer cores que combinem entre si é uma tarefa mais fácil para *designers*. Por isso, existem disponíveis paleta de cores as quais podemos utilizar, o que mitiga esse tipo de dificuldade.

Na realidade, as paletas de cores no R são vetores de cores normalmente na forma hexadecimal. Então, o que precisamos fazer é estabelecer a quantidade de cores que queremos e o estilo da paleta. O estilo da paleta será a combinação de cores que combinam entre si, geralmente um degradê entre duas ou mais cores. Usaremos a função `hcl.colors()` para gerar o vetor de cores da paleta. Abaixo está um exemplo de 5 cores geradas no estilo de paleta "Pastel 1".

```
#gerando paleta de cores
hcl.colors(n = 5, palette = "Pastel 1")
[1] "#FFC5D0" "#E2D4A8" "#A8E1BF" "#A4DDEF" "#E4CBF9"
```

É essa sequência de cores na forma hexadecimal que o R precisa para pôr e embelezar o gráfico. Para listar as paletas possíveis dentro do pacote grDevices (nativo do R), rode a função hcl.pals(). Veja os primeiros nomes:

```
hcl.pals()

[1] "Pastel 1"     "Dark 2"   "Dark 3"    "Set 2"      "Set 3"
[6] "Warm"         "Cold"     "Harmonic"  "Dynamic"    "Grays"
[11]"Light Grays"  "Blues 2"  "Blues 3"   "Purples 2"  "Purples 3"
[16]"Reds 2"       "Reds 3"   "Greens 2"  "Greens 3"   "Oslo"
(…)
```

Algumas paletas são mais fáceis de entender através dos seus nomes, como “Light Grays” (cinzas claros), “Reds 2” (tons de vermelho), “Greens 2” (tons de verde), etc. Já outras teremos que descobrir testando mesmo, pois o nome não nos ajuda muito, como “Set 2”, “Set 3”, “Oslo”, enfim. O blog do Karlo Guidoni nos ajuda nessa tarefa. Acesse https://karlo-guidoni.com/post/2020-07-26-paletas-de-cores-disponiveis-no-r/ e verá o uso de diversas paletas do pacote grDevices.

Como sugestão, teste as paletas: "Blue-Red", "Zissou 1", "Viridis", "Lajolla", "YlOrRd", "ag_Sunset", "Plasma" e "Oslo".

Vamos criar um banco de dados fictício com uma variável categórica do tipo fator e outra numérica do tipo contínua, e testar a paleta “Viridis” em um gráfico de barras.

```
#montando banco de dados
set.seed(071)
dados <- data.frame(
  Categoria = factor(paste("Categ", 1:10),
                     levels = paste("Categ", 1:10)),
  Valor = runif(10, min = 1, max = 100)
)

#Figura 19.52
dados %>%
  ggplot(aes(x = Categoria, y = Valor, fill = Categoria)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = hcl.colors(n = 10, palette = "Viridis"))
```

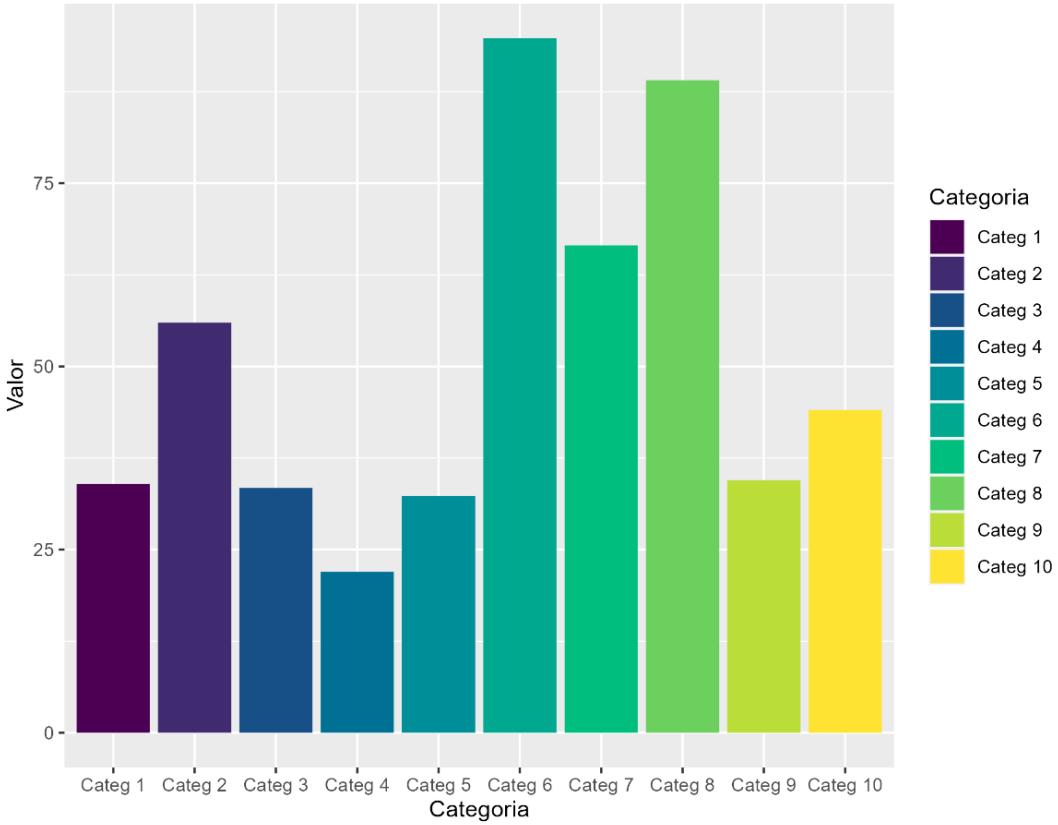


Figura 19.52. Gráficos de barras com paleta de cores do tipo Viridis.

Perceba que a função hcl.colors() foi posta como argumento da função de escala de preenchimento, pois o intuito é mudar as cores internas de cada barra. Mas isso não é obrigado. Há pessoas que gostam de criar um objeto com a paleta desejada e depois colocar o nome desse objeto como argumento. Vamos fazer isso modificando o estilo para "Lajolla".

```
#Figura 19.53
paleta <- hcl.colors(n = 10, palette = "Lajolla")

dados %>%
  ggplot(aes(x = Categoria, y = Valor, fill = Categoria)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = paleta)
```

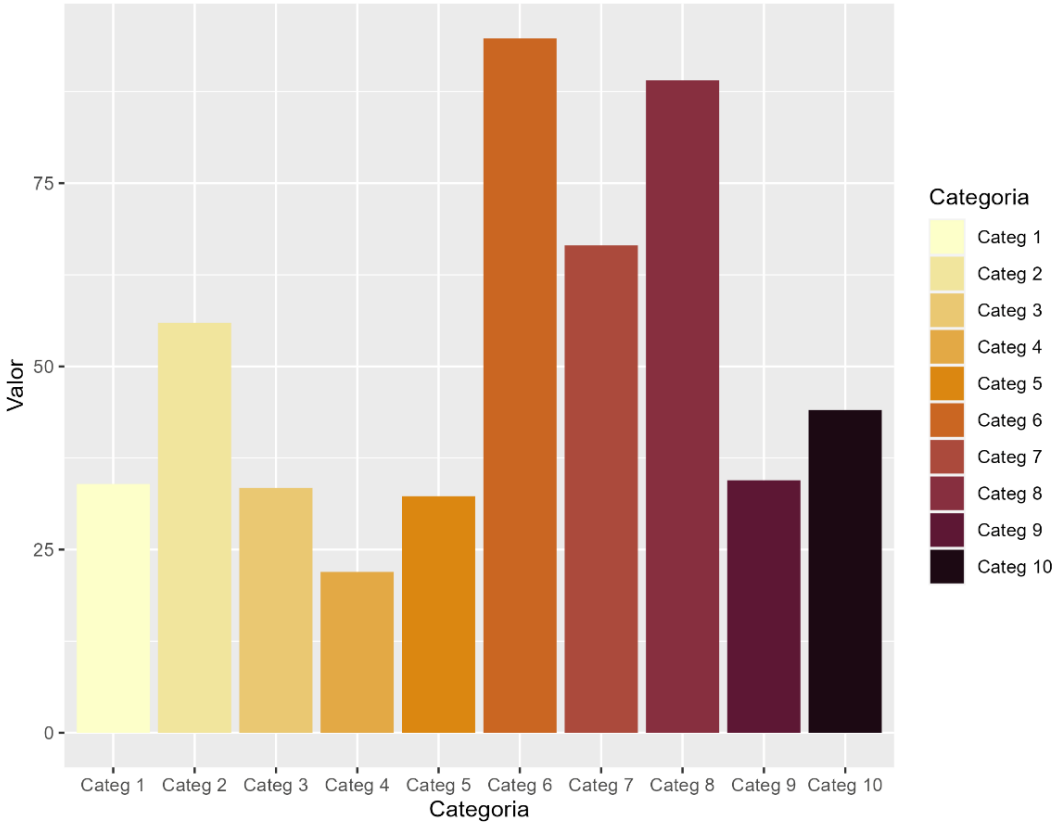


Figura 19.53. Construindo paleta de cores com hcl.colors().

Sendo assim, você pode criar a sua própria paleta de cores, basta criar um vetor com a função de combinar c() e listar dentro dela as cores desejadas.

19.2.19 Temas

O ggplot2 traz consigo diversos temas pré-definidos. Para alterar o tema do gráfico, basta adicionar ao código as funções que começam com theme_ que mostraremos a seguir. Após o sublinhado deve vir o nome do tema desejado. Por exemplo, o tema padrão do ggplot2, o qual vimos em diversos gráficos acima é o theme_gray() ou theme_grey(), que é o tema com tons de cinza.

Abaixo está a lista de todos os temas possíveis, usando o ggplot2, na Figura 19.54.

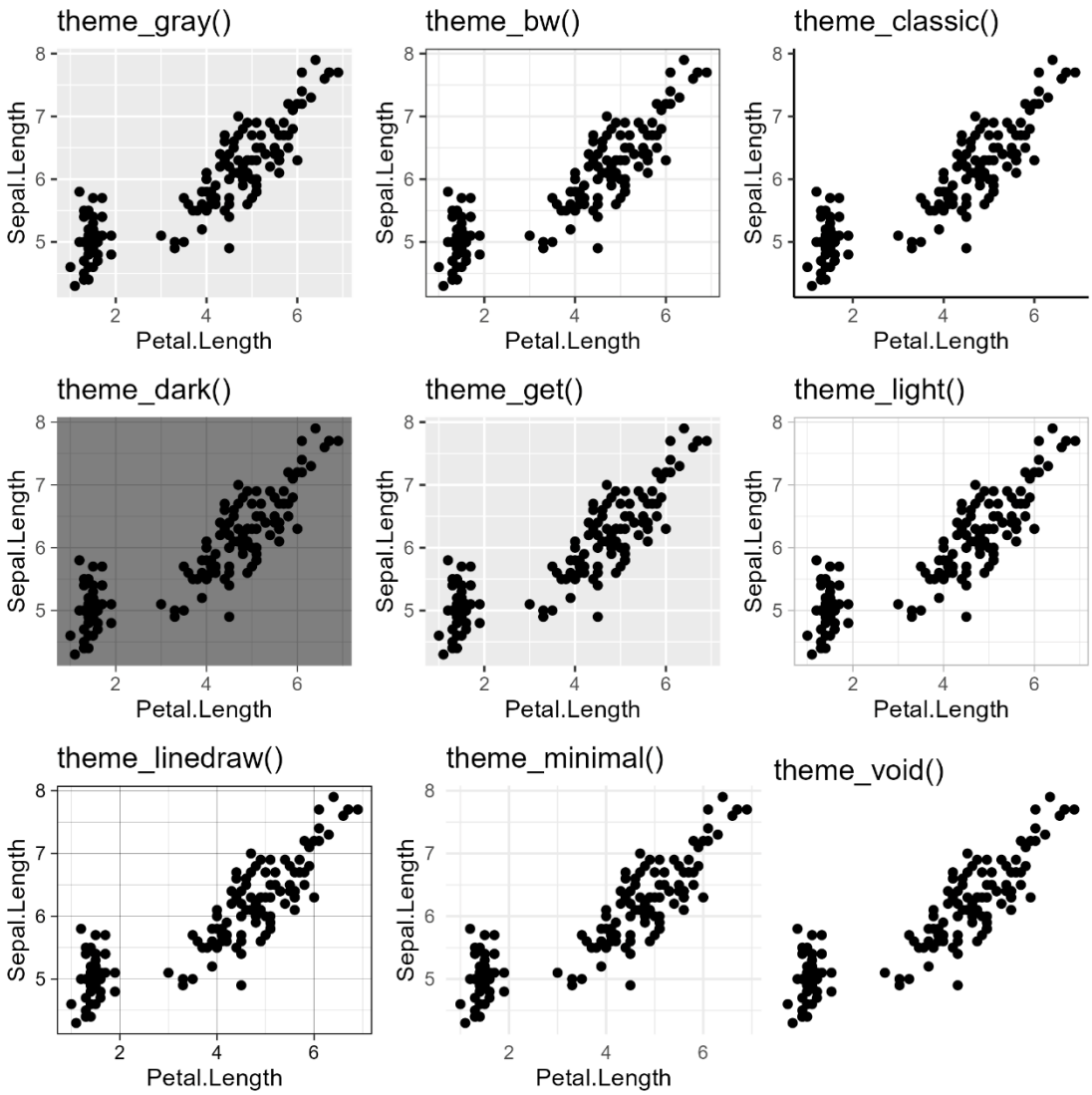


Figura 19.54. Temas nativos do ggplot.

Dentre esses temas, falando-se de artigo científico ou outro trabalho que exija rigor na apresentação visual, os temas *classic* e *minimal* são os mais adequados, pois são mais "limpos". Veja agora o código para mudar o tema para o *classic*, por exemplo.

```
#Figura 19.55
iris %>%
  ggplot(aes(x = Petal.Length, y = Sepal.Length)) +
  geom_point() +
  theme_classic()
```

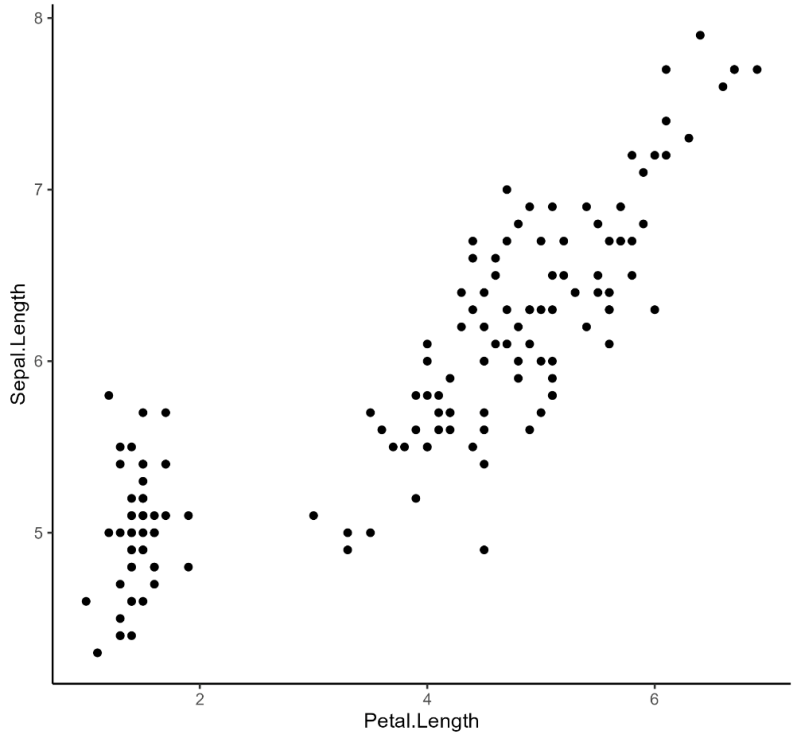


Figura 19.55. gráfico de pontos com tema clássico.

Existe um outro pacote acessório chamado ggthemes que além de gerar paletas de cores, escalas e outras funções mais, podemos encontrar outros temas que podem ser utilizados em nossos gráficos gerados pelo ggplot2. As funções referidas aos temas são:

- theme_base()
- theme_calc()
- theme_clean()
- theme_economist()
- theme_economist_white()
- theme_excel()
- theme_excel_new()
- theme_few()
- theme_fivethirtyeight()
- theme_foundation()
- theme_gdocs()
- theme_hc()
- theme_igray()
- theme_map()
- theme_pander()
- theme_par()
- theme_solarized()
- theme_solarized_2()
- theme_solid()
- theme_stata()
- theme_tufte()
- theme_wsj()

Há dois temas desse pacote que acredito serem bastante úteis. Vou exemplificar o uso do ggthemes com eles. No primeiro utilizaremos a função theme_stata(). Esse tema se torna interessante para quem trabalha com o programa estatístico STATA e não quer perder os padrões de tema.

```
install.packages('ggthemes')
library(ggthemes)
```

```
#Figura 19.56
iris %>%
  ggplot(aes(x = Petal.Length, y = Sepal.Length)) +
  geom_point() +
  theme_stata()
```

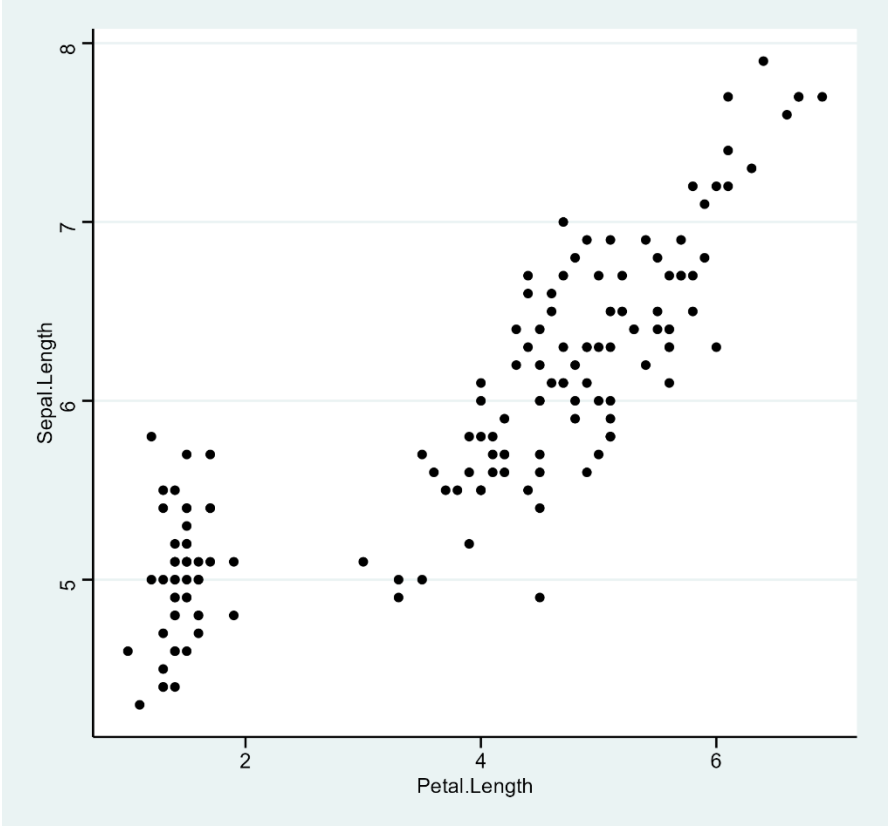


Figura 19.56. Tema STATA do pacote ggthemes.

No exemplo abaixo, usando theme_par() se torna interessante se você necessitar misturar gráficos do pacote graphics (nativo do R), que são muito simples, com outros do ggplot. Com essa função o tema será semelhante entre eles.

```
#Figura 19.57
iris %>%
  ggplot(aes(x = Petal.Length, y = Sepal.Length)) +
  geom_point() +
  theme_par()
```

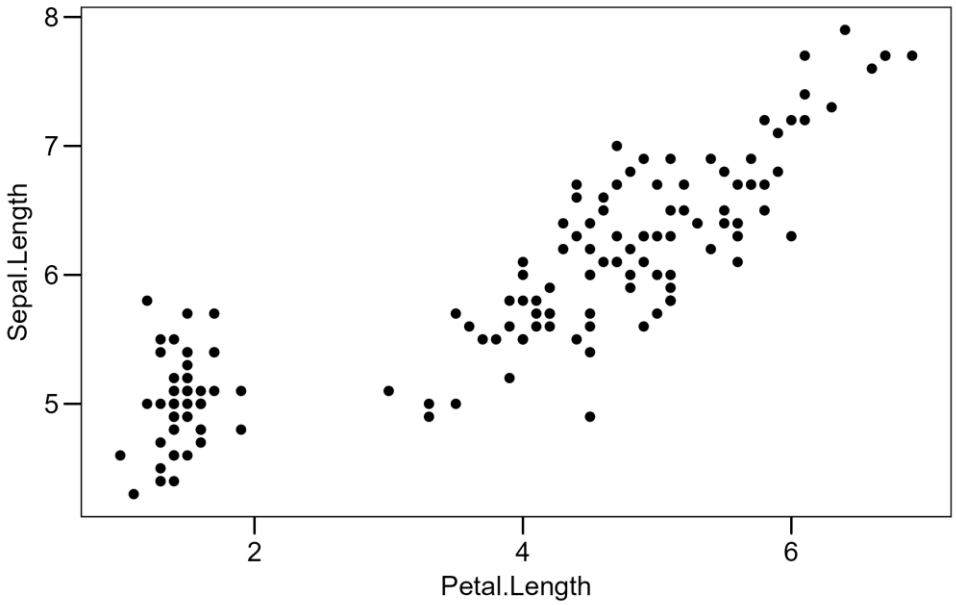


Figura 19.57. Tema "par", que se assemelha ao tema padrão do pacote graphics, nativo do R.

Para ver mais temas usando o pacote ggthemes, entre outras funcionalidades, acesse o site All Your Figure Are Belong To Us, em https://yutannihilation.github.io/allYourFigureAreBelong-ToUs/ggthemes/.

19.2.20 Facetas (múltiplos gráficos)

Facetar no ggplot2 é na realidade construir múltiplos gráficos com base em uma variável categórica ou inteira. Para realizar essa proeza incrível, podemos utilizar duas funções: facet_grid() ou face_wrap(). A diferença entre elas é:

- facet_grid(): você escolhe facilmente se a divisão dos gráficos vai ser em linhas ou colunas. Uma característica da função é que os gráficos gerados ficam alargados, seja verticalmente ou horizontalmente.
- facet_wrap(): os gráficos gerados ficam mais quadrados.

Uma característica comum entre ambos é que a geometria (ou seja, o estilo do gráfico) será igual entre todos os gráficos. Então, se escolhermos construir gráficos de pontos, será gráfico de pontos para todos; se for escolhermos a geometria de *boxplot*, será *boxplot* para todos os gráficos. No próximo subcapítulo veremos como juntar em uma figura gráficos de geometrias diferentes.

Para exemplificar, vamos adicionar uma variável contínua a mais no banco de dados que já conhecemos: ToothGrowth. Vamos pôr uma variável hipotética para a largura dental, em milímetros, usando a função runif(). Veja como:

```
data(ToothGrowth)

set.seed(222)
#largura dental em milímetros
ToothGrowth$width <- runif(1:60, min = 3, max = 6)

ToothGrowth
    len supp dose    width
1   4.2   VC  0.5 5.794778
2  11.5   VC  0.5 3.199084
3   7.3   VC  0.5 4.497736
4   5.8   VC  0.5 3.000778
5   6.4   VC  0.5 5.749091
6  10.0   VC  0.5 5.872166
7  11.2   VC  0.5 4.055680
8  11.2   VC  0.5 4.259473
9   5.2   VC  0.5 4.719140
10  7.0   VC  0.5 3.433980
11 16.5   VC  1.0 4.207483

(…)
```

Um gráfico de pontos criado com todos os dados seria feito usando o código abaixo:

```
#Figura 19.58
ToothGrowth %>%
  ggplot(aes(x = len, y = width)) +
  geom_point()
```

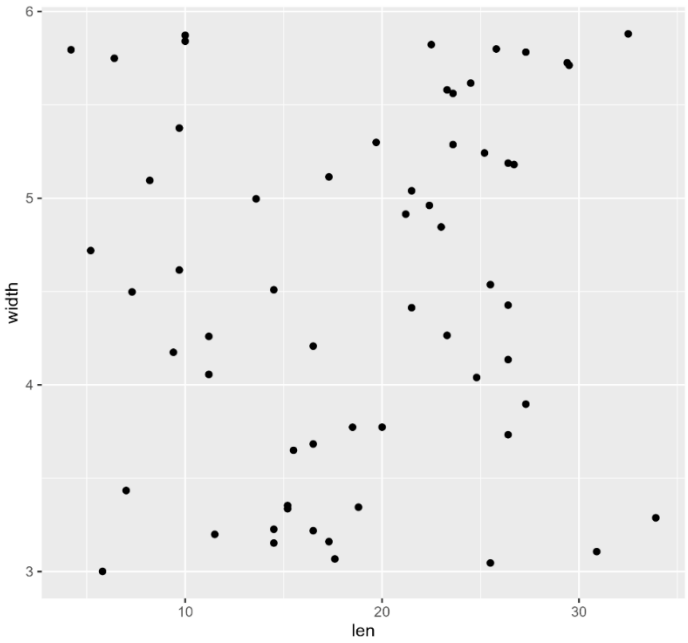


Figura 19.58. Gráfico de pontos com todos os dados.

Entretanto, vamos imaginar que precisemos construir gráficos de pontos diferentes para cada classe da variável supp (tipo de suplementação: VC [vitamina C] ou OJ [suco de laranja]). Usando a função facet_grid(), veja como os gráficos ficam alargados na horizontal quando estabelecemos a divisão por linha (*rows*):

```
#Figura 19.59
ToothGrowth %>%
  ggplot(aes(x = len, y = width)) +
  geom_point() +
  facet_grid(rows = vars(supp))
```

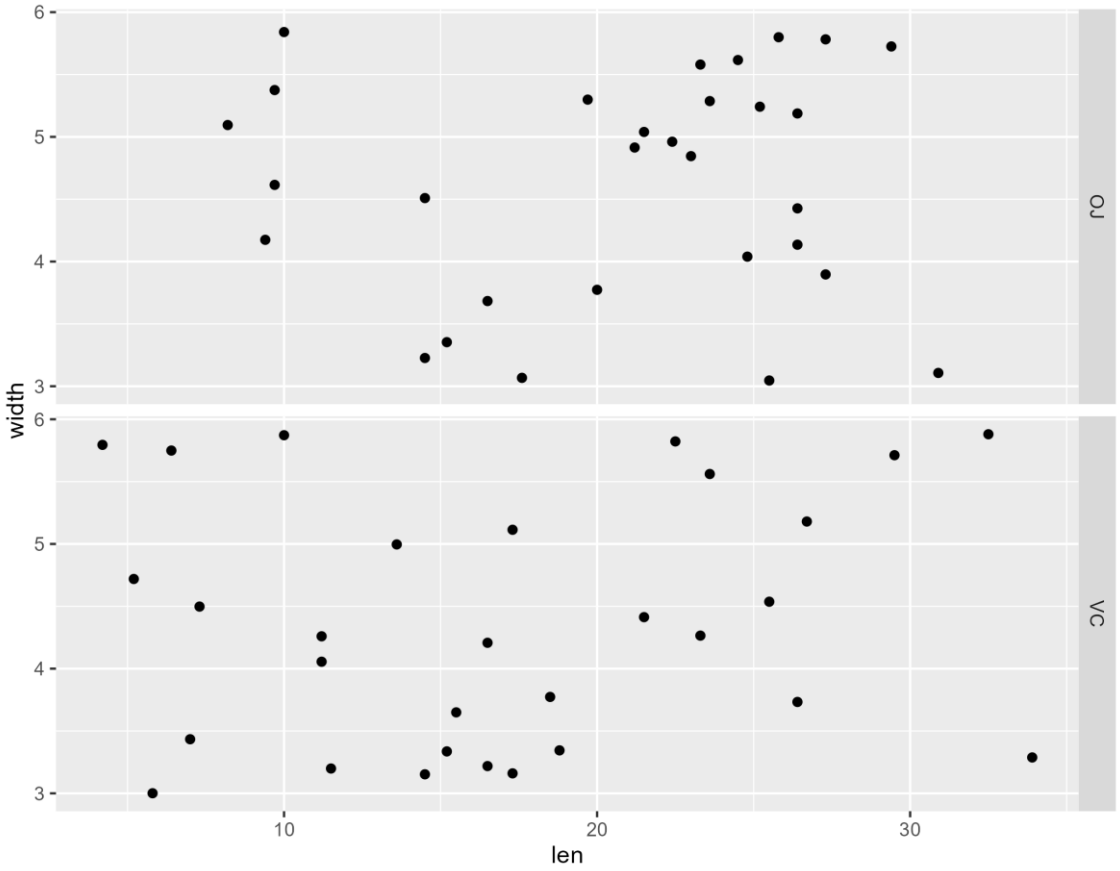


Figura 19.59. Gráficos de pontos separados por grupos, horizontalmente.

Essa formatação é interessante para gráficos de linhas, quando expressão uma série temporal. Agora, veja como fica por colunas:

```
#Figura 19.60
ToothGrowth %>%
  ggplot(aes(x = len, y = width)) +
  geom_point() +
  facet_grid(cols = vars(supp))
```

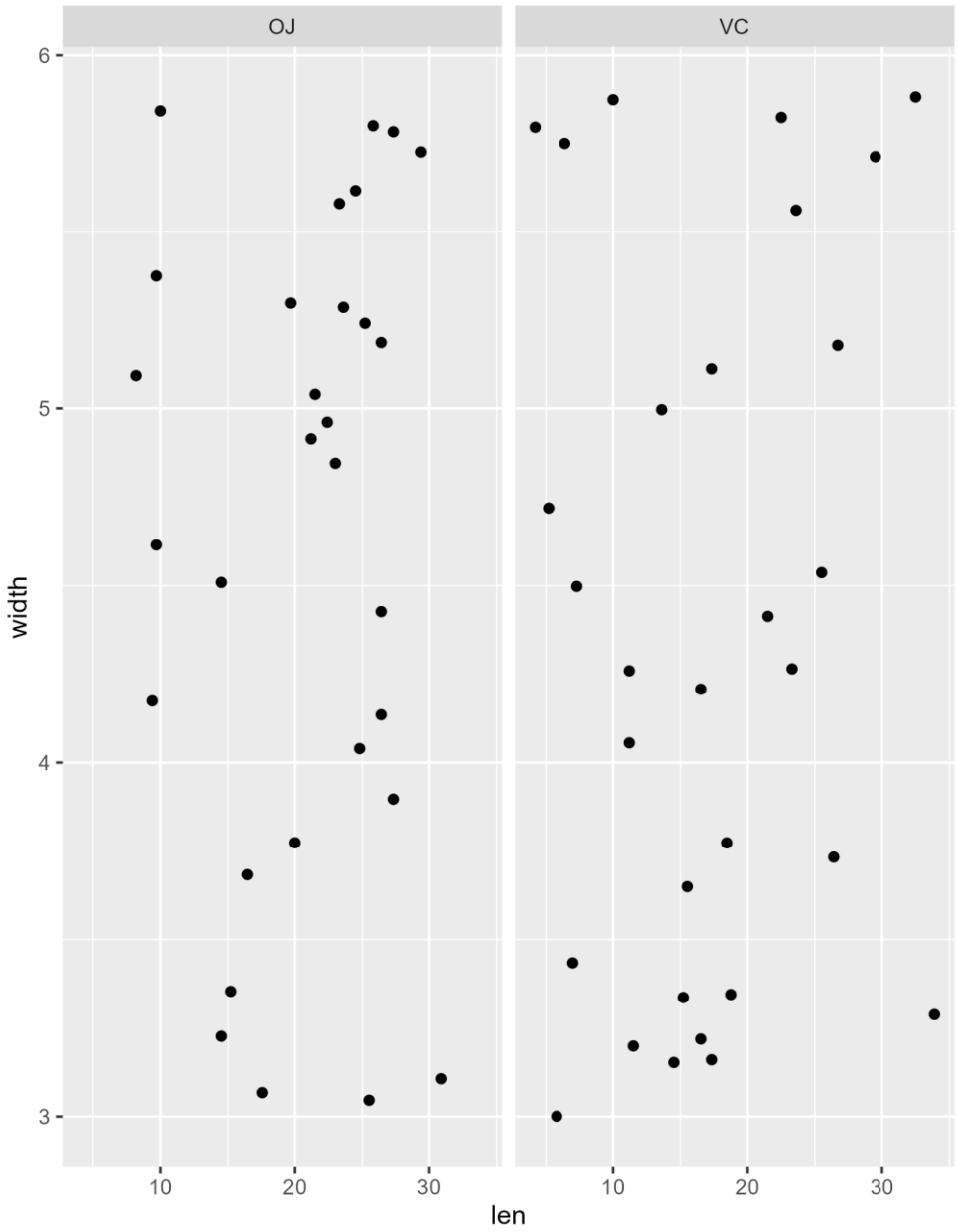


Figura 19.60. Gráficos de pontos separados por grupos, verticalmente.

Podemos estabelecer duas variáveis categóricas diferentes entre linhas e colunas, e assim subdividiremos ainda mais os gráficos. Perceba:

```
#Figura 19.61
ToothGrowth %>%
  ggplot(aes(x = len, y = width)) +
  geom_point() +
  facet_grid(rows = vars(dose),
             cols = vars(supp))
```

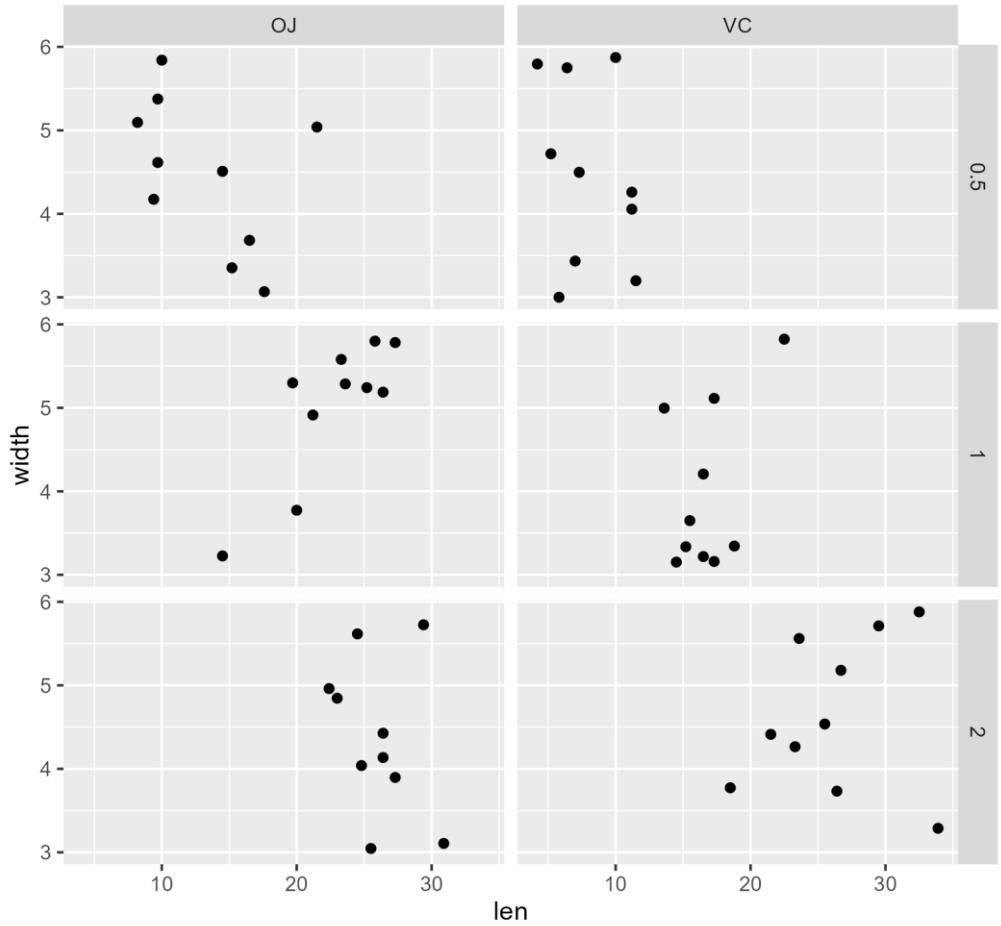


Figura 19.61. Gráficos de pontos separados por grupos entre linhas e colunas.

Para exemplificar o uso com facet_wrap(), podemos em vars() estabelecer mais de uma variável para separar os gráficos. No próximo exemplo, vamos dividir novamente por suplementação e dose. Veja como fica nessa situação:

```
#Figura 19.62
ToothGrowth %>%
  ggplot(aes(x = len, y = width)) +
  geom_point() +
  facet_wrap(vars(supp, dose))
```

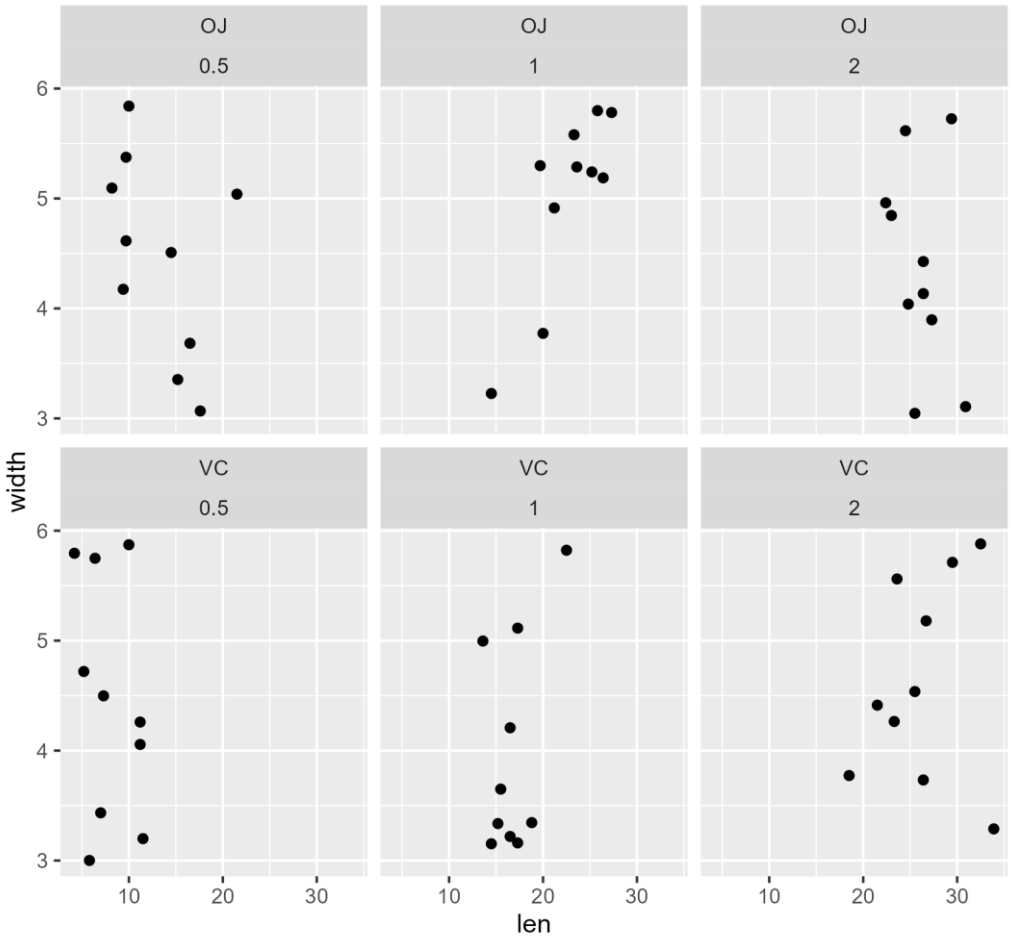


Figura 19.62. Gráficos de pontos separados por dois grupos subdivididos somente em colunas.

Um parâmetro interessante em facet_wrap() é "ncol", onde podemos estabelecer o número de colunas na figura final. Vamos estabelecer que a figura fique com apenas duas colunas. Isso pode ser útil quando o espaço no documento de escrita é restrito ou ficar mais adequado para o entendimento do leitor (que não é esse caso, pois o gráfico ficará ruim de interpretar). Perceba:

```
#Figura 19.63
ToothGrowth %>%
  ggplot(aes(x = len, y = width)) +
  geom_point() +
  facet_wrap(vars(supp, dose), ncol = 2)
```

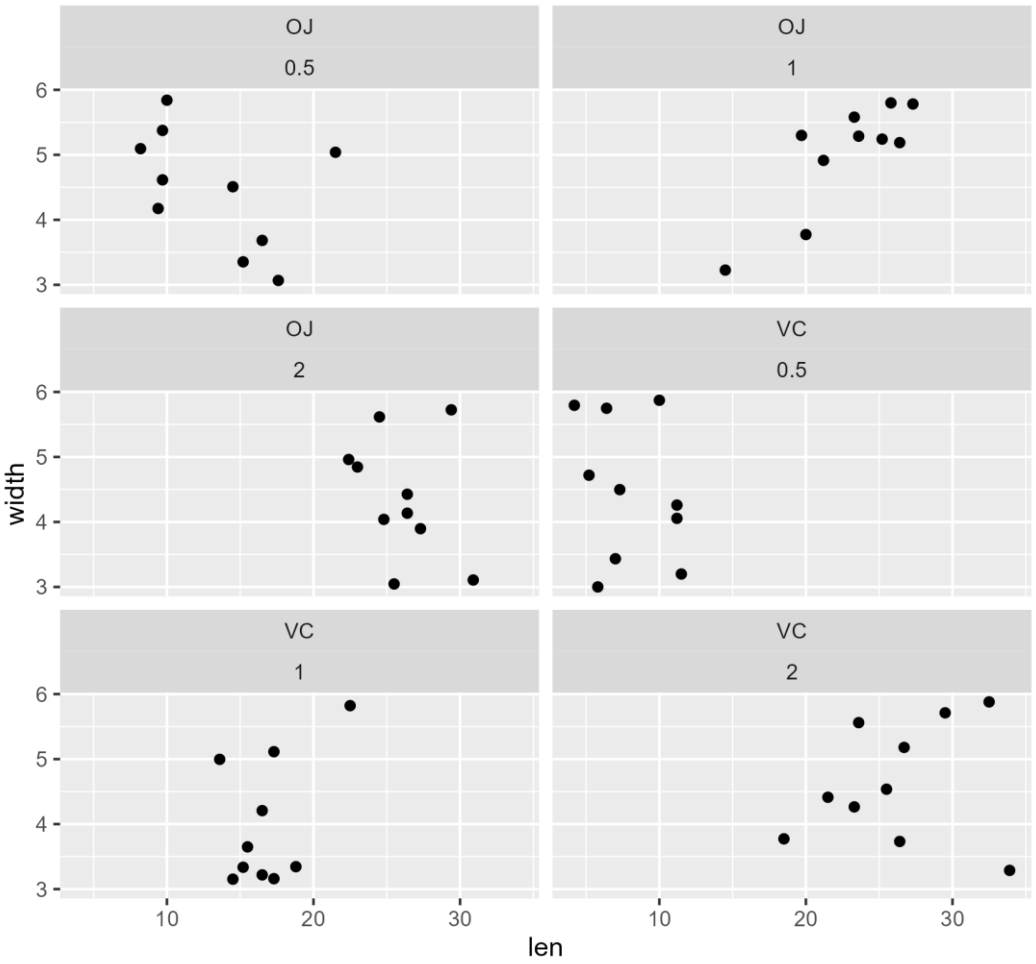


Figura 19.63. Definindo gráfico subdivididos em duas colunas.

No site de referências do ggplot2 (https://ggplot2.tidyverse.org/reference/) veja mais exemplos sobre facetas.

19.2.21 Múltiplos gráficos de tipos diferentes

Para juntar vários gráficos diferentes criados via ggplot é possível usando pacotes específicos para isso. Apresentarei aqui o que acredito ser o mais fácil de montar a figura final: o pacote patchwork (traduzindo: colcha de retalhos).

Tal pacote usa funções simples da aritmética como soma e divisão para formular a figura final. Quando "somamos" gráficos, indica que estamos adicionando-os à figura. Quando "dividimos" gráficos, indica que queremos pôr um acima do outro. Vai ser muito mais fácil demonstrar do que tentar te explicar em palavras. Vamos instalar o patchwork, construir quatro gráficos diferentes e montar figuras mais complexas a partir deles.

```
install.packages('patchwork')
library(patchwork)
library(tidyverse)

g1 <- ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point()

g2 <- ggplot(iris, aes(Sepal.Length, fill = Species)) +
  geom_density(alpha = 0.5)

g3 <- ggplot(iris, aes(Sepal.Length, Sepal.Width, fill = Species)) +
  geom_boxplot()

g4 <- ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_line()
```

Agora que temos salvos os quatro gráficos em objetos diferentes, podemos usá-los para construir a figura final. De g1 a g4, verifique que respectivamente se trata de um gráfico de pontos, gráfico de densidade, um boxplot e um gráfico de linhas. Primeiro, vamos adicionar os dois primeiros gráficos, um do lado do outro.

```
#Figura 19.64
g1 + g2
```

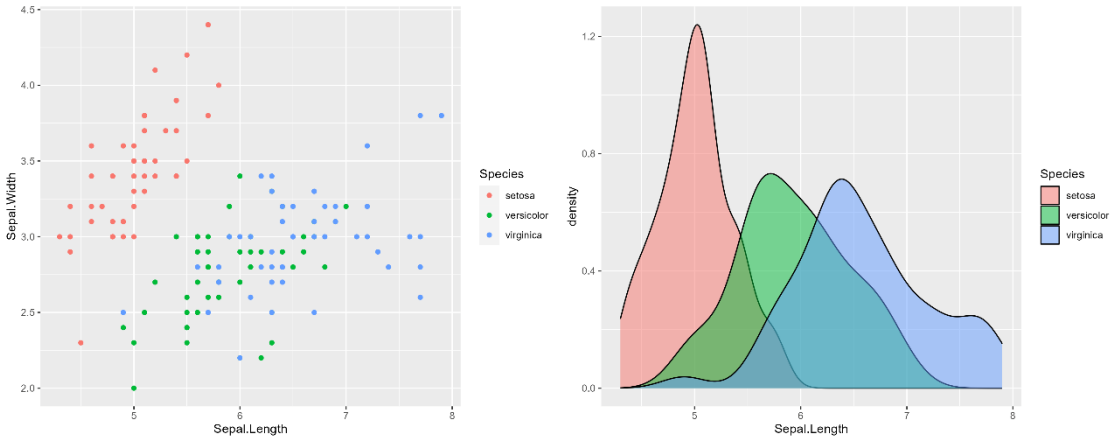


Figura 19.64. Gráficos distintos em uma linha, formando uma só figura.

Vamos agora pôr um acima do outro.

```
#Figura 19.65
g1 / g2
```

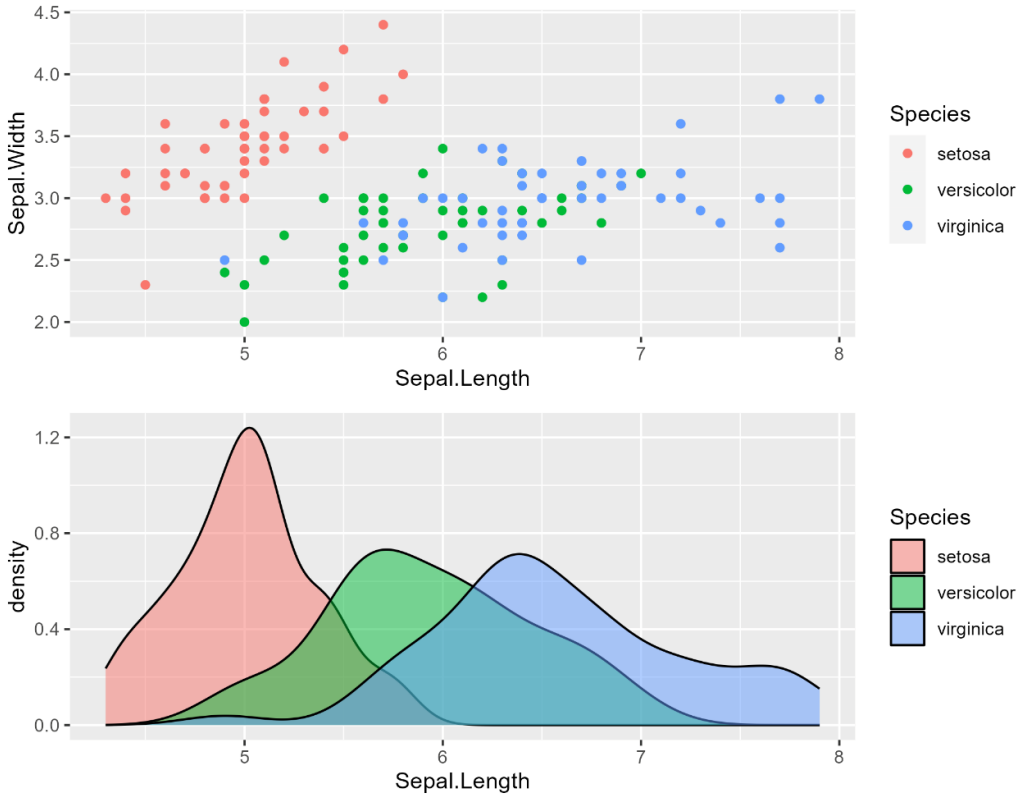


Figura 19.65. Gráficos distintos em duas linhas, formando uma só figura.

O ajuste do comprimento do gráfico pode ser feito na hora de salvar a imagem (veremos mais à frente) ou quando você clicar no botão de zoom do RStudio e ajustar o tamanho da janela.

Vamos pôr dois gráficos em cima e só um abaixo:

```
#Figura 19.66
(g1 + g2) / g3
```

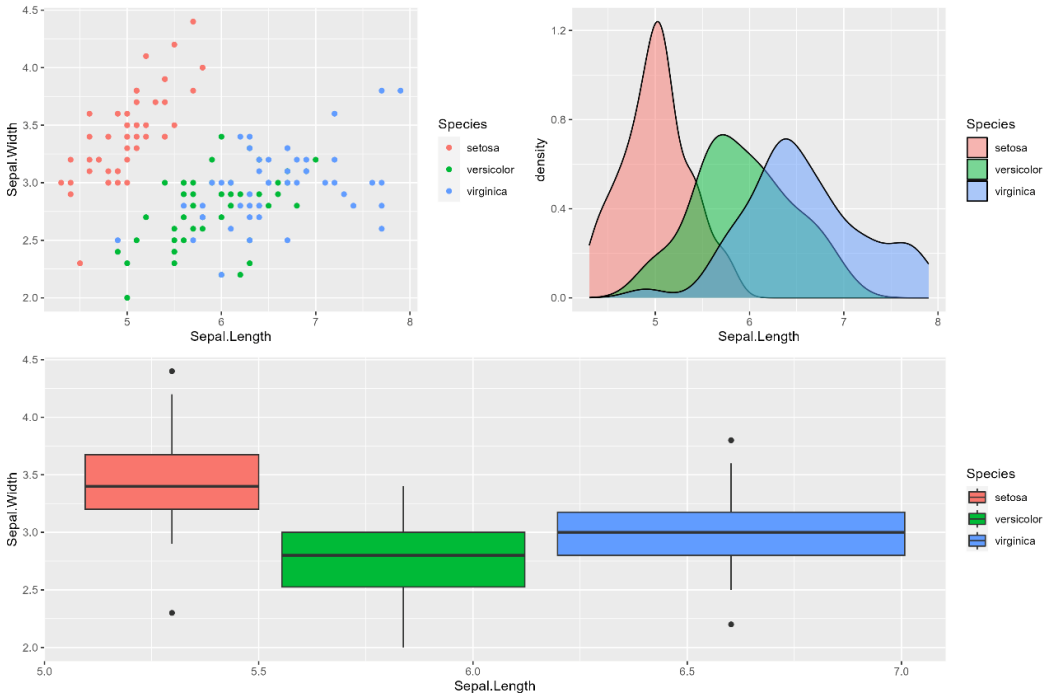


Figura 19.66. Gráficos distintos entre linhas e colunas, formando uma só figura.

Vamos adicionar agora os quatro gráficos:

```
#Figura 19.67
g1 + g2 + g3 + g4
```

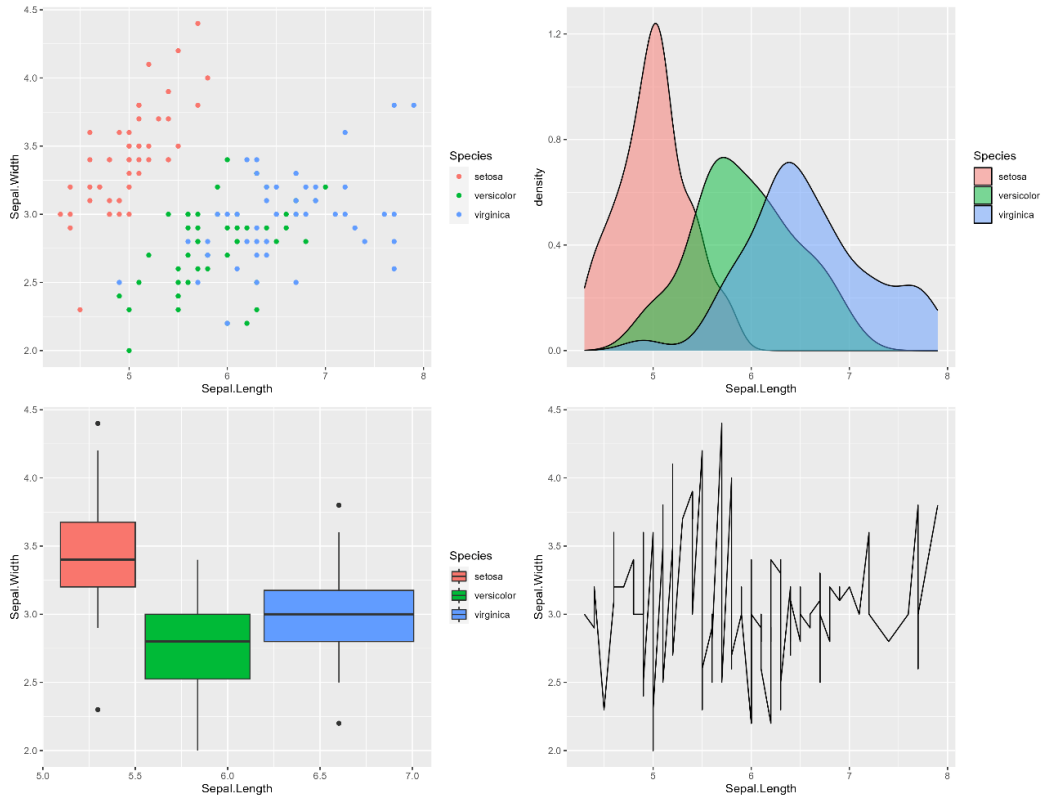


Figura 19.67. Quatro gráficos distintos em uma só figura.

Perceba que o patchwork distribuiu em duas linhas e duas colunas. Mas, e se eu quisesse distribuir os gráficos em uma só coluna? Para isso precisamos estabelecer o número de colunas ou linhas em uma função acessória chamada plot_layout(). São parâmetros interessantes dessa função para mexer no *layout* da figura final:

- nrow = estabelece o número de linhas;
- rcol = estabelece o número de colunas;
- widths = estabelece a largura dos gráficos entre as colunas ("quero uns gráficos mais largos que outros");
- heights = estabelece a altura dos gráficos entre as linhas ("quero uns gráficos com maior altura do que outros");
- design = você pode "desenhar" como você quer a distribuição dos gráficos na figura final.

Veja como fica a figura quando colocamos todos os gráficos em uma coluna só:

```
#Figura 19.68
g1 + g2 + g3 + g4 + plot_layout(ncol = 1)
```

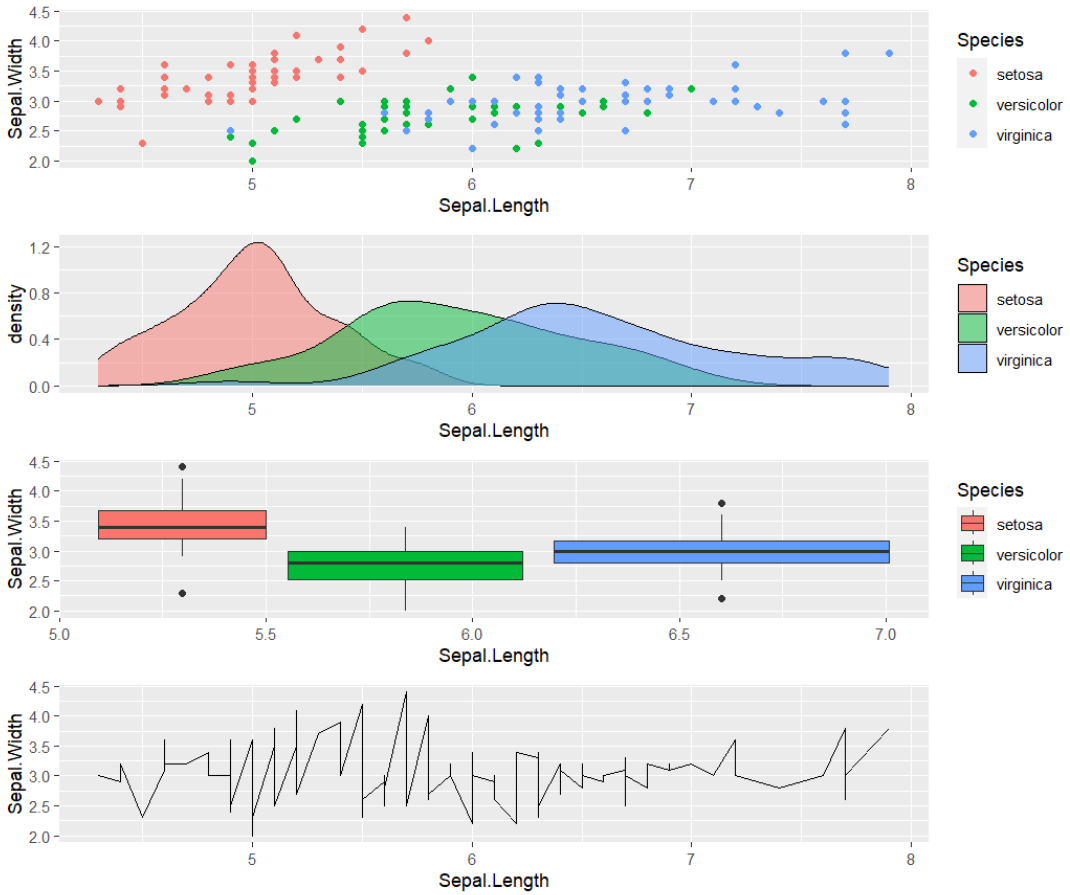


Figura 19.68. Quatro gráficos distintos dispostos em uma só coluna.

No comando abaixo estamos pedindo os quatro gráficos, mas que no layout os gráficos que ficarem na primeira coluna tenham uma largura 1x (o tamanho original) e que os gráficos que ficarem na segunda coluna tenha uma largura 2x (o dobro do tamanho). Veja como fica:

```
#Figura 19.69
g1 + g2 + g3 + g4 + plot_layout(widths = c(1,2))
```

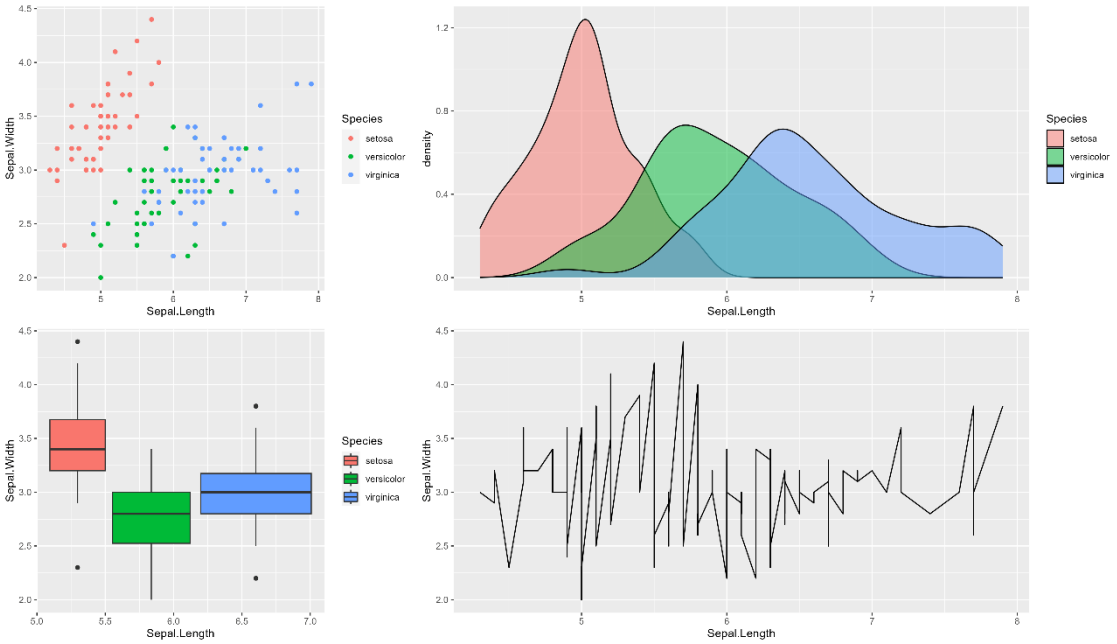


Figura 19.69. Quatro gráficos distintos com *layout* modificado.

Podemos "desenhar" nossa figura final. Para isso, devemos gerar um esboço em um vetor colocando números que serão referência para cada gráfico (1, 2, 3, 4, 5, etc.). Quando repetimos o número, isso indicará a área tomado por aquele gráfico. O caractere "#" deve ser adicionado quando quisermos deixar um espaço vazio. Veja o exemplo:

```
#Figura 19.70
meu_desenho <- "
  122
  1#3
  #44
"
g1 + g2 + g3 + g4 + plot_layout(design = meu_desenho)
```

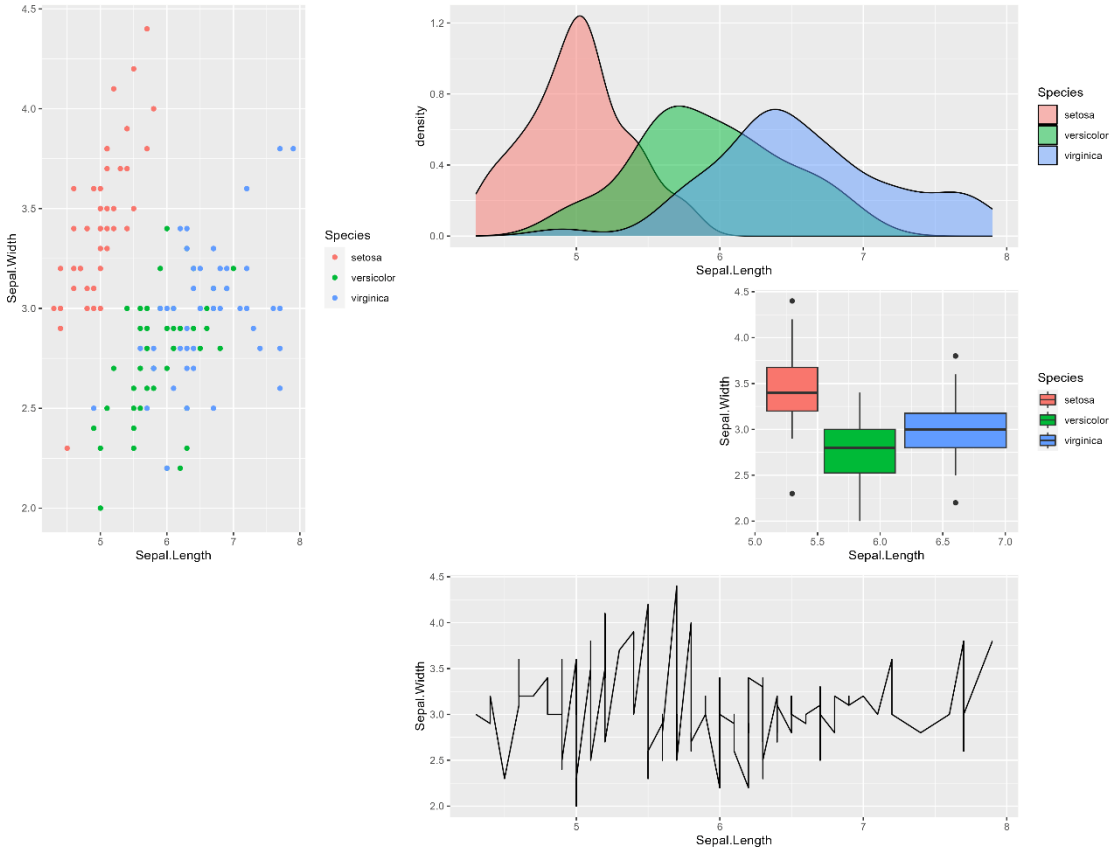


Figura 19.70. Gráficos distintos com *layout* desenhado previamente.

Podemos acrescentar um espaço vazio também usando a função plot_spacer(). Veja exemplo:

```
#Figura 19.71
(g1 + g2) / (plot_spacer() + g3)
```

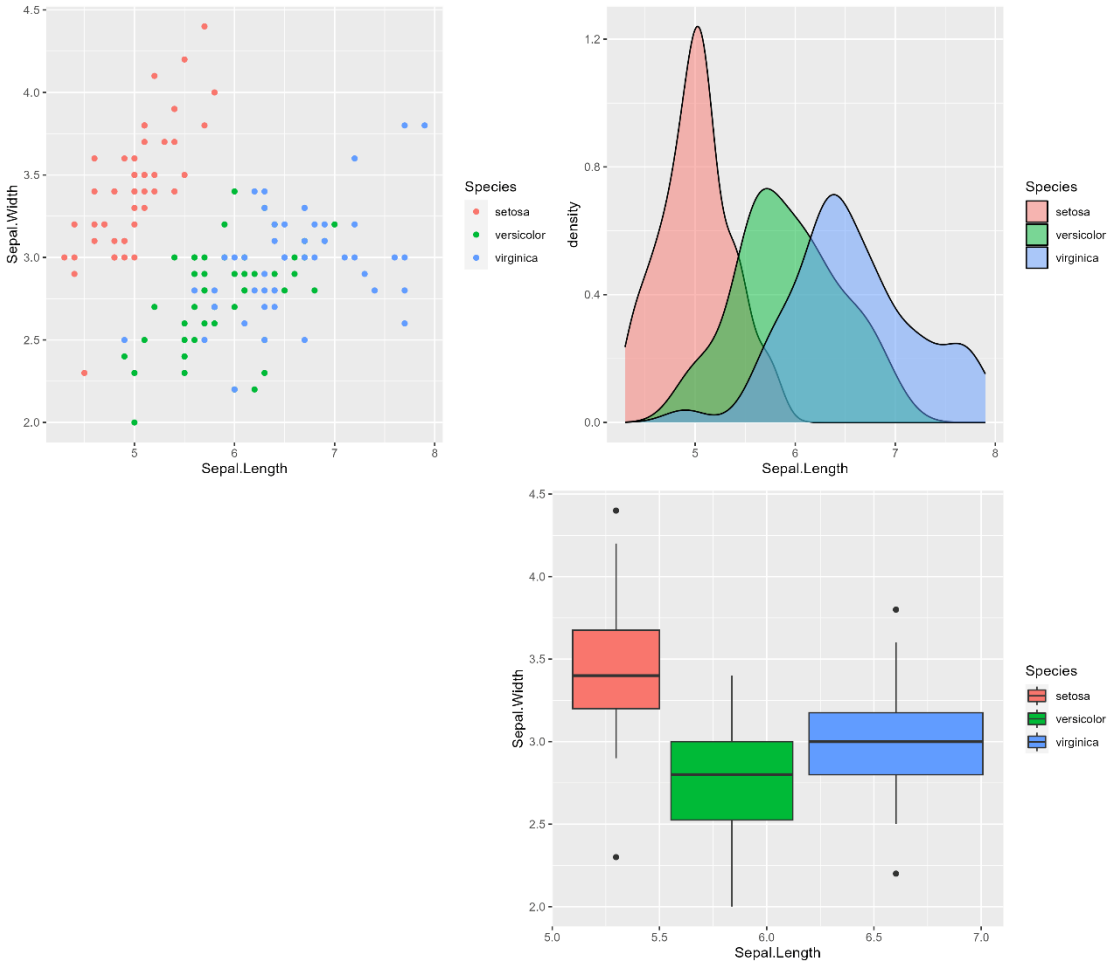


Figura 19.71. Acrescentando espaços vazios na figura final.

Caso você não tenha se adaptado ao patchwork ou só quer desbravar outras possibilidades, podemos utilizar outros pacotes para fazer a função de juntar gráficos vindos do ggplot2 em uma só figura. Busque por gridExtra ou cowplot para expandir as possibilidades.

19.2.22 Transformando gráficos do pacote graphics em ggplot

Ao longo do livro, trabalhamos também com gráficos que não foram gerados pelo ggplot2. Lembrando que as funções `plot()`, `boxplot()`, `barplot()`, `lines()`, `hist()` e outras pertencem ao pacote graphics que é nativo do R. Caso você deseje juntar gráficos feitos com o ggplot2 e outros feitos pelo graphics, você não conseguirá. Dará o seguinte erro:

```
g5 <- barplot(1:10)
g1 + g5
Error in `ggplot_add()`:
```

```
! Can't add `g5` to a <ggplot> object.
Run `rlang::last_trace()` to see where the error occurred.
```

O erro afirma que não consegue adicionar g5 como objeto do ggplot. Ou seja, o patchwork só juntará gráficos feitos pelo ggplot2.

Para contornar esse problema, podemos usar o pacote ggplotify que transforma gráficos do graphics em objeto ggplot, e aí sim podemos juntar gráficos com o patchwork. Instale e abra a biblioteca.

```
install.packages('ggplotify')
library(ggplotify)
```

É com a função as.ggplot() que podemos transformar o gráfico em um objeto ggplot. Dentro da função, como argumento, devemos pôr o código do gráfico vindo do graphics. Vamos primeiro construir um gráfico de barras simples, meramente demonstrativo.

```
#Figura 19.72
#gráfico de barras original do graphics
barplot(1:10)
```

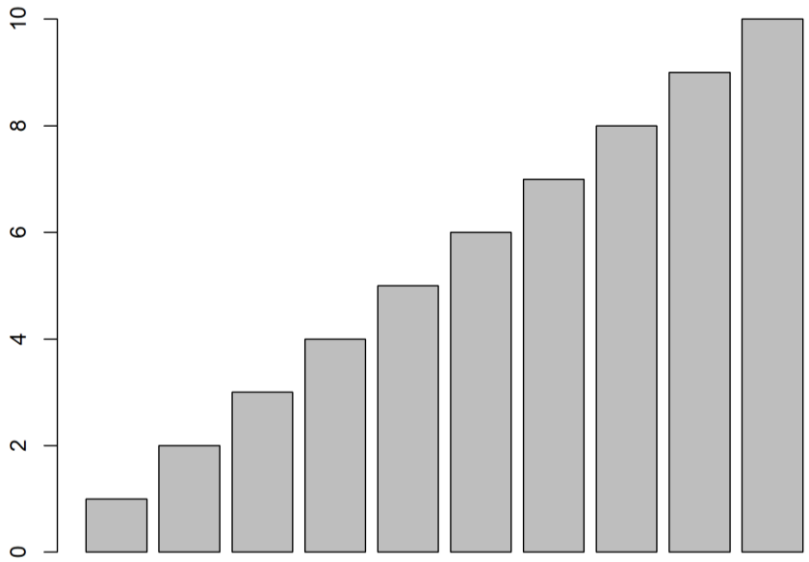


Figura 19.72. Gráfico de barras simples criado pelo pacote graphics.

```
#Figura 19.73
#transformando em objeto do ggplot
g5 <- as.ggplot(~barplot(1:10))
g5
```

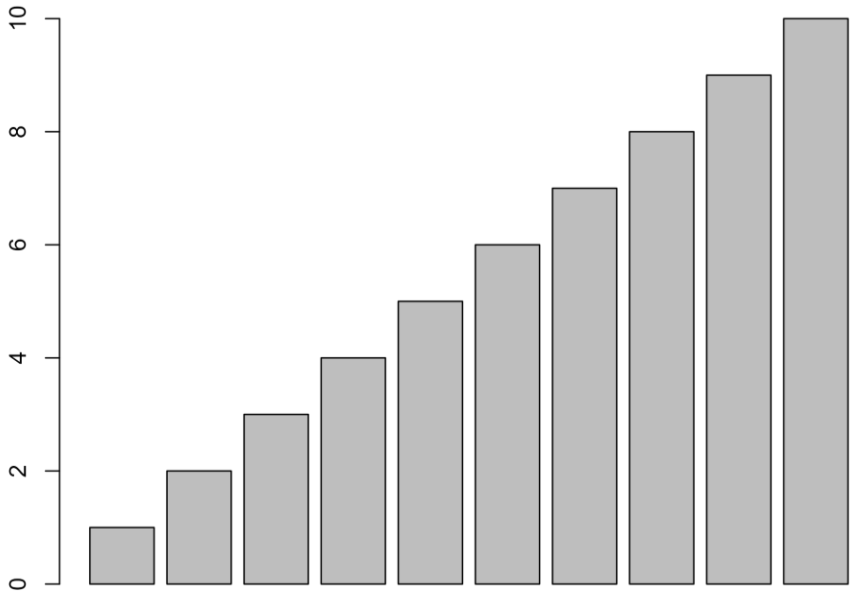


Figura 19.73. Gráfico de barras do graphics transformado em objeto ggplot.

Nesse momento você pode dizer "Não mudou nada". Visualmente é verdade e é isso que eu queria te mostrar, mas no último comando o gráfico será agora um objeto do tipo ggplot. Com isso entendido, vamos juntá-los em uma única figura utilizando o pacote patchwork que estudamos acima.

```
#Figura 19.74
g1 + g5
```

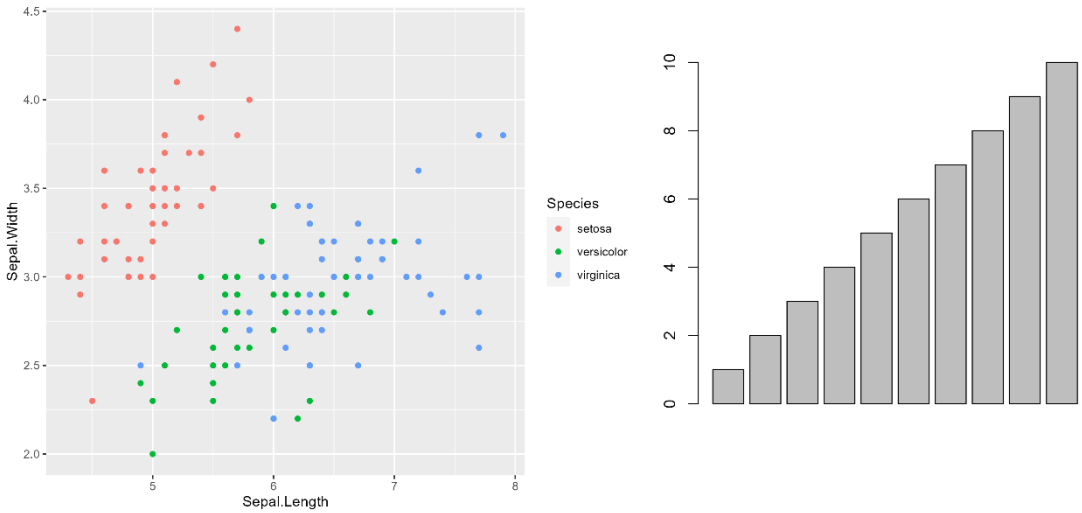


Figura 19.74. Gráficos de pacotes originalmente de pacotes diferentes postos em uma só figura.

19.2.23 Sobrepondo gráficos diferentes

Já mexemos aqui com sobreposições de gráficos diferentes ou de elementos gráficos, como retas e textos. Vamos fechar esse conhecimento aqui para que esse recurso seja utilizado da melhor forma possível e que não gere erros que venham atravancar a sua experiência com o R.

Primeiramente, devemos criar gráficos sobrepostos usando o mesmo banco de dados. Então, usaremos apenas uma vez a função ggplot(), local onde deve ficar estipulado o nome do banco. As geometrias, ou seja, os tipos de gráficos é que estarão de acordo com as funções geom_ que se deseja adicionar.

A sequência de funções geom_ é que indicará ao R os níveis de camadas da imagem. Vamos ver o gráfico abaixo e comentar sobre sua complexidade.

```
#Figura 19.75
ToothGrowth %>%
  ggplot(aes(x=as.factor(dose), y=len, fill = as.factor(dose))) +
  geom_violin(trim = F, alpha = 0.2) +
  geom_boxplot() +
  geom_jitter()
```

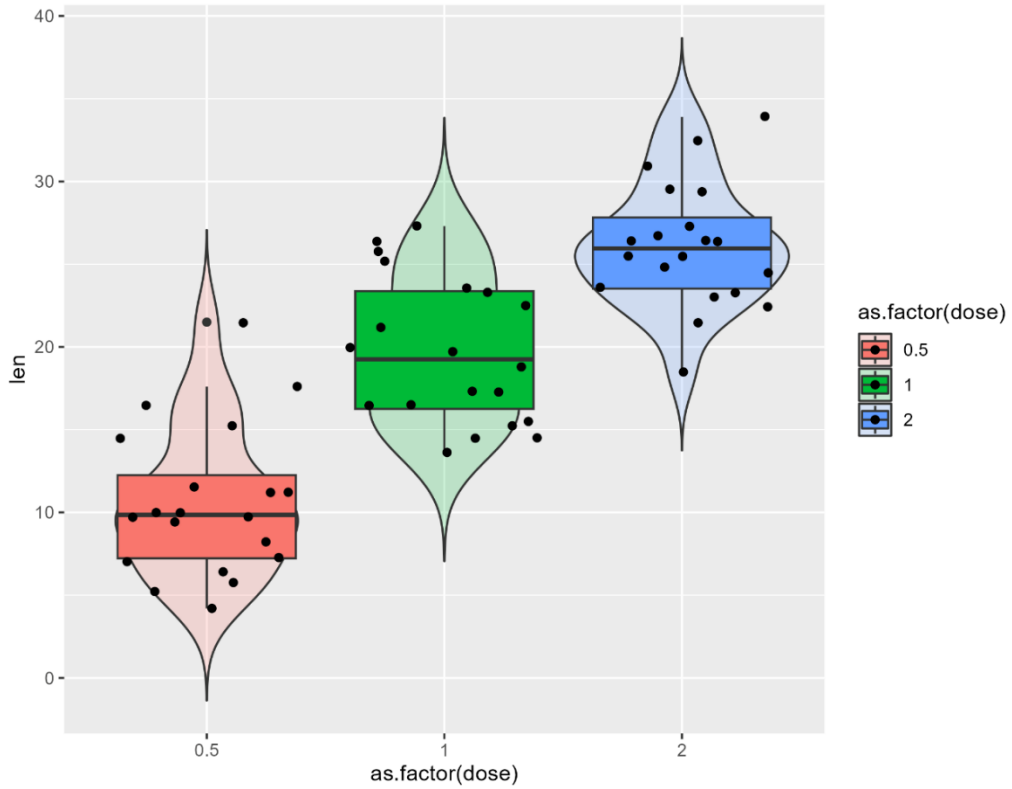


Figura 19.75. Gráficos violino sobrepostos a *boxplots*, sobrepostos a gráficos *jitter*.

A primeira coisa a ser observada é que o banco ToothGrowth está vinculado com *pipe* (%>%) à função ggplot(). Poderíamos escrever o banco dentro da própria função e o sentido seria exatamente o mesmo.

Depois, nós escrevemos a estética geral do gráfico dentro da função ggplot(), lembrando que isso não é obrigatório e veremos abaixo que podemos escrever estéticas diferentes dentro de cada função geom_. Aqui, a dose está no eixo x e colocamos do tipo fator. No eixo y está a variável len (comprimento do dente). O preenchimento está também de acordo com a dose, mas poderia ser qualquer variável categórica.

Depois, vemos três camadas, ou seja, as sobreposições gráficas. Em uma figura só, temos três gráficos diferentes: o primeiro e mais ao fundo são gráficos violino, o segundo ou intermediário são *boxplots*, e os últimos e mais superficiais na imagem são gráficos do tipo *jittered points*. Perceba que nos gráficos violinos foi estabelecida uma transparência com "alpha", para que haja percepção na diferença entre os gráficos.

Vejamos outro exemplo agora, com o código e a Figura 19.76 gerada abaixo:

```
#Figura 19.76
ggplot(iris) +
  geom_point(aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point(aes(x = Petal.Width, y = Petal.Length), color = "red") +
  geom_boxplot(aes(x = Sepal.Length), alpha = 0.5) +
  geom_boxplot(aes(y = Petal.Length), alpha = 0.5)
```

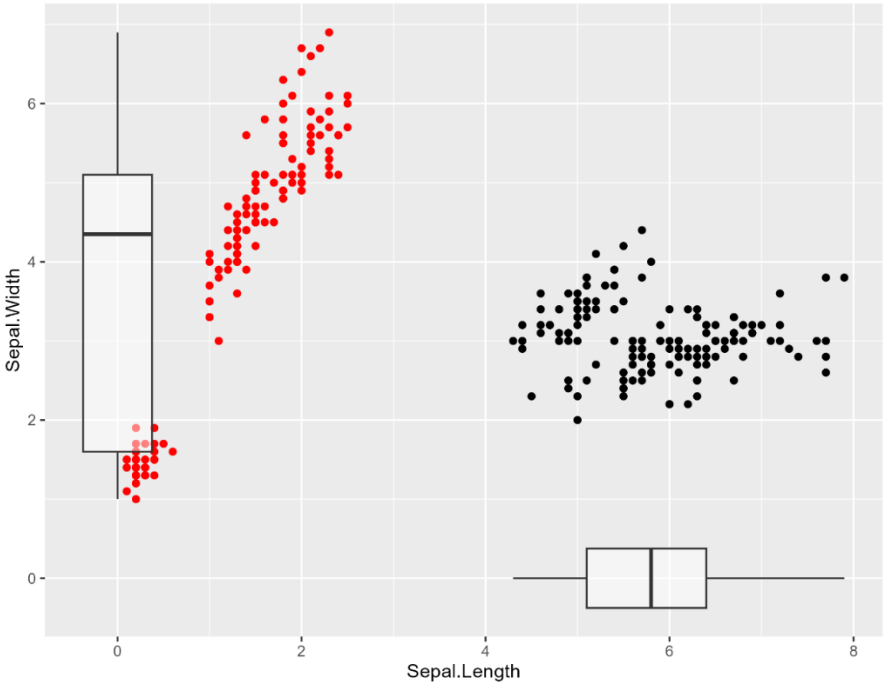


Figura 19.76. Sobrepondo *boxplot* aos gráficos de pontos e avaliando visualmente distribuições de dados.

Nessa figura perceba que o *pipe* não foi utilizado e o banco foi estipulado dentro de ggplot(). As duas primeiras camadas são dois gráficos de pontos com variáveis x e y invertidas, e poderiam ser completamente diferentes. Mas nesse ponto o analista precisar ter cautela em relação às escalas entre os gráficos sobrepostos. Se as escalas dos dados numéricos forem muito diferentes (exemplo, uma variável do eixo x varia entre 0 e 1 e a do segundo gráfico varia entre 0 e 100) um dos gráficos será prejudicado e ficará difícil de interpretar.

Perceba que as estéticas (aes()) foram descritas em cada função geom_. Isso foi muito importante neste exemplo, porque as definições dos eixos x e y são diferentes.

A terceira e a quarta camada são *boxplots*, com transparência de 50%. Perceba que o primeiro *boxplot* (na vertical) acabou escondendo alguns pontos da camada anterior. Se não fosse posta a transparência, não daria para vê-los completamente, o que é ruim, pois seria informação perdida.

Outro ponto a ver é que o título dos eixos foi posto com base na primeira camada (o primeiro gráfico de pontos), mas não generaliza a outros seguintes. Dá para mudar isso usando a função labs() ou xlab() e ylab(). Mas, perceba que é preciso planejar toda essa estrutura gráfica, para que falsas informações não sejam passadas a seu público.

19.2.24 Mapas

Construir mapas no R é uma tarefa das mais difíceis partindo do zero. Como não temos tempo hábil para isso, alguns pacotes nos ajudam e promovem interação com o ggplot2, o que facilita e muito a nossa vida. A complexidade dos mapas surge com os dados que devem ser imputados. Um mapa é fundamentalmente um gráfico de polígonos e para isso é preciso ter estabelecido os pontos e linhas que cortam os limites entre países, estados ou mesmo municípios.

Esses dados fundamentais são chamados de dados vetoriais. É muito comum para quem trabalha com geociências utilizar esse tipo de arquivo, o qual chamamos de *shapefiles*, que têm extensão *.shp. Esses arquivos são distribuídos na internet facilmente. O Instituto Brasileiro de Geografia e Estatística (IBGE), por exemplo, cuida da formação e atualização dos arquivos vetoriais de todo o território brasileiro. Acesse www.ibge.gov.br, a parte de Downloads, bases cartográficas contínuas, caso tenha interesse nisso futuramente.

Mas, para a nossa salvação, há pacotes do R que já têm dados vetorizados de diversas partes do mundo. Então, não vamos nos preocupar aqui com *shapefiles*. No primeiro momento, o próprio

ggplot2 concederá os dados de vetores mundiais para trabalharmos com o mapa global. Para "puxar" esses dados de vetores, usaremos a função map_data(), e nela estabeleceremos o banco "world" como argumento. Esse mapa chamado "world" terá como centro o Oceano Atlântico. Se você desejar ter o Oceano Pacífico como centro, escolha "world2". Vamos salvar esses dados em um objeto que chamaremos de *world* também.

Para construir um genérico mapa-múndi precisamos utilizar além da função ggplot(), a geometria poligonal, como dito acima, com a função geom_polygon(). Como o globo é dividido em linhas imaginárias chamadas de latitude (normalmente horizontais) e longitude (normalmente verticais), devemos considerar isso na geometria do gráfico. Além do mais, temos que estabelecer argumento para o parâmetro "group", que tem grande importância na construção do mapa. Ele controla (entre outras coisas) se os pontos adjacentes devem ser conectados por linhas. Se estiverem no mesmo grupo, eles se conectam, mas se estiverem em grupos diferentes, não serão conectados.

Em resumo, teremos vários pontos de latitude e longitude que delimitarão os limites de um país, por exemplo, e todos esses pontos devem pertencer a um só grupo, que formará a área do próprio país. É esse tipo de dado que o ggplot2 entenderá e utilizará para construir os polígonos de cada país, estado ou município.

Vamos abrir nosso banco de vetores mundiais:

```
library(tidyverse)
world <- map_data("world")

head(world)
       long      lat group order region subregion
1 -69.89912 12.45200     1     1  Aruba      <NA>
2 -69.89571 12.42300     1     2  Aruba      <NA>
3 -69.94219 12.43853     1     3  Aruba      <NA>
4 -70.00415 12.50049     1     4  Aruba      <NA>
5 -70.06612 12.54697     1     5  Aruba      <NA>
6 -70.05088 12.59707     1     6  Aruba      <NA>
```

Perceba que esse banco de dados tem em sua primeira coluna os valores de longitude (long); na segunda os de latitude (lat); na terceira coluna os valores de grupo; *order*, que é a ordem dos pontos; *region*, que neste caso está descrito o nome dos países ou de uma região específica do globo (vemos acima Aruba); e, *subregion* são sub-regiões, quando houver.

Vamos construir nosso primeiro mapa. O objeto *world* entrará como banco de dados, a longitude como parâmetro do deixo x, a latitude como parâmetro do eixo y, e o *group* no parâmetro

"group". Vamos remover a legenda atribuindo "none" ao parâmetro "legend.position", na função theme().

```
#Figura 19.77
#mapa-múndi simples
ggplot(world) +
  geom_polygon(aes(x = long, y = lat, group = group)) +
  theme(legend.position = "none")
```

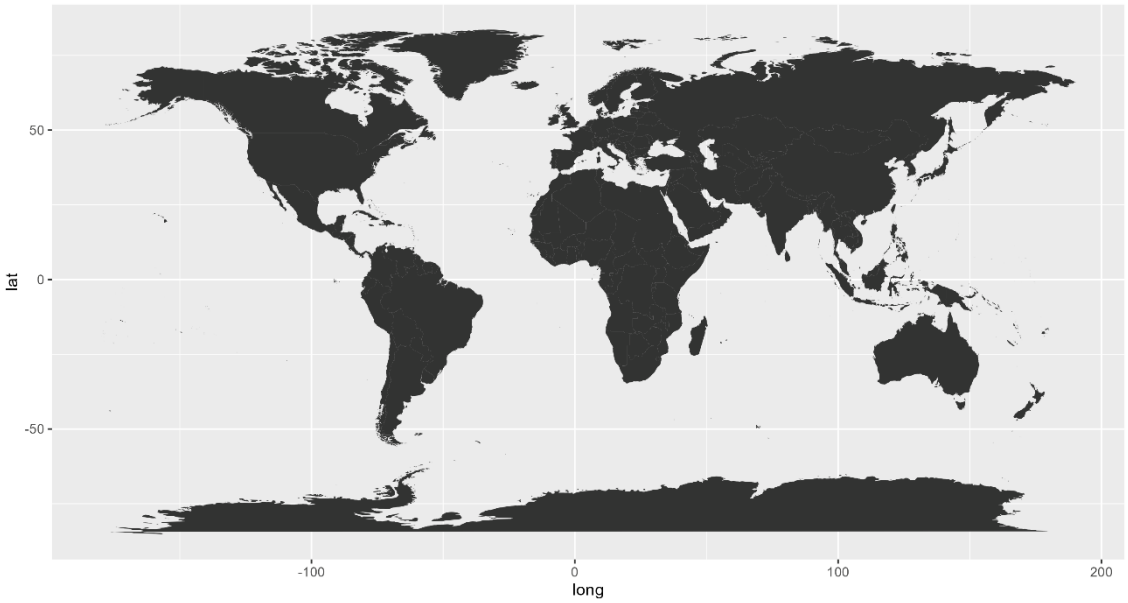


Figura 19.77. Mapa-múndi com base no ggplot.

Como o mapa criado se trata de um objeto do ggplot, podemos facilmente manipular as escalas dos eixos com scale_x_continuos() e scale_y_continuos(). Vamos mexer em *breaks* (quebras), mas lembrando você pode também estabelecer os limites em *limits*. Além disso, vamos mudar nesse próximo exemplo o tema e o tamanho da fonte das escalas dos eixos. Veja:

```
#Figura 19.78
#mudando escala e mudando o tema
ggplot(world) +
  geom_polygon(aes(x = long, y = lat, group = group)) +
  scale_x_continuous(breaks = seq(-180, 180, 30)) +
  scale_y_continuous(breaks = seq(-90, 90, 10)) +
  theme_minimal() +
  theme(legend.position = "none",
        axis.text = element_text(size = 20))
```

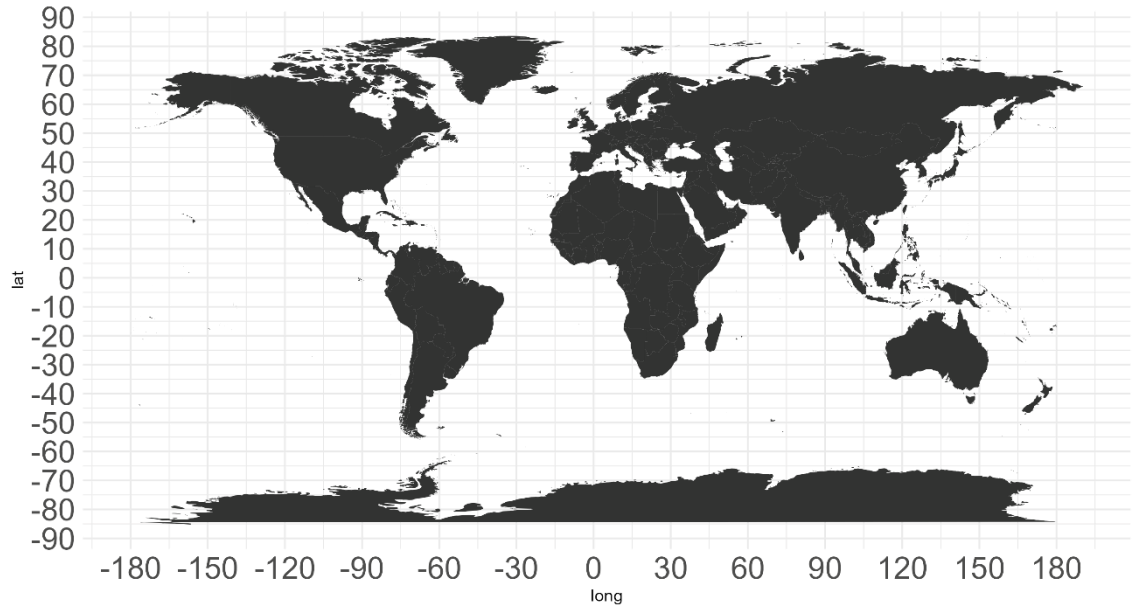


Figura 19.78. Modificação das escalas e tema do mapa.

Como a variável *region* está descrito os nomes dos países, podemos colorir cada um deles usando preenchimento (*fill*) na estética do gráfico.

```
#Figura 19.79
#cada país tem uma cor
ggplot(world) +
  geom_polygon(aes(x = long, y = lat, group = group, fill = region)) +
  theme(legend.position = "none")
```

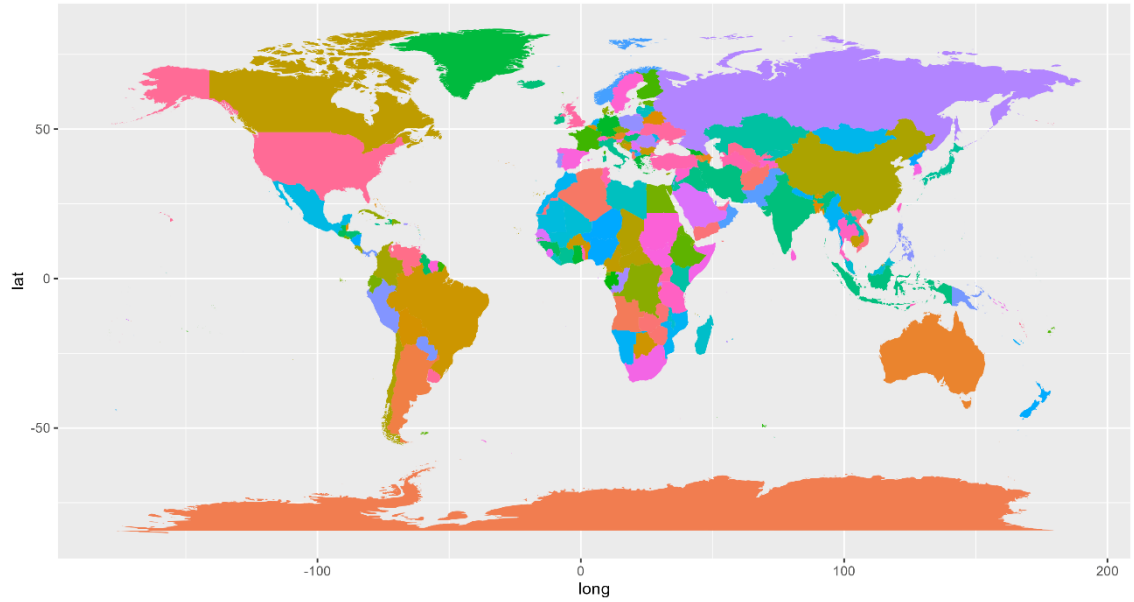


Figura 19.79. Modificação das cores de cada país no mapa.

Esse gráfico colorido do jeito que está não é muito informativo, pois são muitas regiões e algumas acabam tendo a mesma cor sem que haja relação entre elas. Mas, nesse ponto, é um bom momento para a gente abordar um tipo de mapa específico.

Mapas nós temos de diversos tipos: mapas climáticos, topográficos, históricos, enfim. Existe um mapa temático em especial que utilizamos muito na área epidemiológica, é o chamado mapa coroplético.

Um mapa coroplético é um tipo de mapa que representa áreas geográficas (como países, estados ou municípios) colorindo-as ou sombreando-as de acordo com uma variável estatística. Essa variável pode ser qualquer dado quantitativo ou qualitativo associado a essas áreas, como quantitativo populacional, densidade demográfica, taxa de desemprego, taxas de prevalência, taxas de incidência, taxas de óbitos, quantitativos de agravos, entre muitos outros exemplos. O objetivo principal é visualizar distribuições e padrões dessas variáveis de forma clara e intuitiva no mapa.

Se você abre o mapa da sua rua pelo seu celular, aquilo não é mapa coroplético. Todos os mapas a seguir sim serão. Vamos, por exemplo, diferenciar em cores alguns países da América Latina, mas antes, devemos construir no banco de dados uma variável que os diferencie. Vou chamar essa variável de "valores" em nosso exemplo. Brasil, Argentina, Chile e Equador estarão com a mesma cor de destaque e terão valor 1, enquanto que os demais países terão uma cor diferente tendo esses o valor zero. Depois vamos construir nosso mapa colocando a variável "valores" como preenchimento. Vamos mudar a cor das linhas de borda entre os países para preto.

```
#Figura 19.80
#lista de países em destaque
paises <- c('Brazil', 'Argentina', 'Chile', 'Ecuador')

#criando variável nova
world$valores <- ifelse(world$region %in% paises, 1, 0)

ggplot(world) +
  geom_polygon(aes(x = long, y = lat, group = group, fill = valores), color =
'black')
```

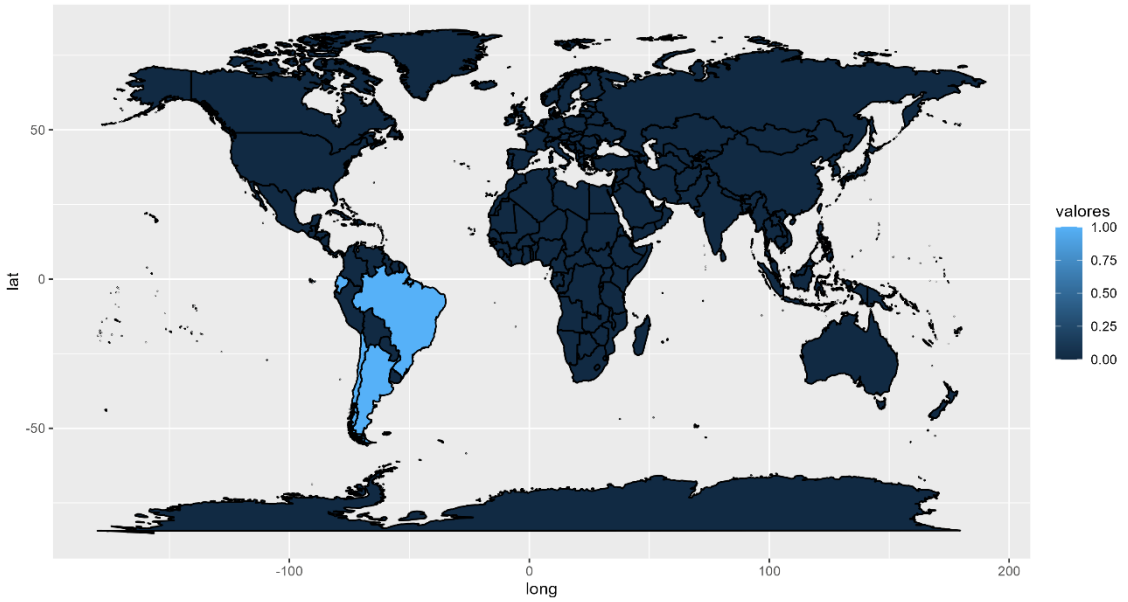


Figura 19.80. Destaque de países específicos no mapa.

Para saber o nome de todas as regiões possíveis do bando "world" ou "world2", basca rodar o comando:

```
unique(map_data('world')$region)
  [1] "Aruba"                   "Afghanistan"
  [3] "Angola"                  "Anguilla"
  [5] "Albania"                 "Finland"
  [7] "Andorra"                 "United Arab Emirates"
  [9] "Argentina"               "Armenia"
 [11] "American Samoa"          "Antarctica"
(...)
```

No exemplo acima demos um destaque a nível qualitativo, apesar de darmos valores a eles. Escolhemos os países e eles têm destaques iguais. Poderia ser os países que têm casos de uma doença específica, por exemplo.

Contudo, vamos além no mapa coropléticos. Vamos ver como ficaria se utilizarmos uma variável numérica do tipo contínua, simulado as taxas de prevalência de alguma doença, por exemplo. Para isso, vamos recriar a variável "valores" para que tenhamos números aleatórios entre 40 e 80. Vamos também remover a linha que delimita os países alterando "linewidth" para zero e ver como ficará o mapa dessa maneira.

```
#Figura 19.81
set.seed(111)
world$valores <- runif(nrow(world), min = 40, max = 80)

ggplot(world) +
  geom_polygon(aes(x = long, y = lat, group = group, fill = valores), linewidth = 0)
```

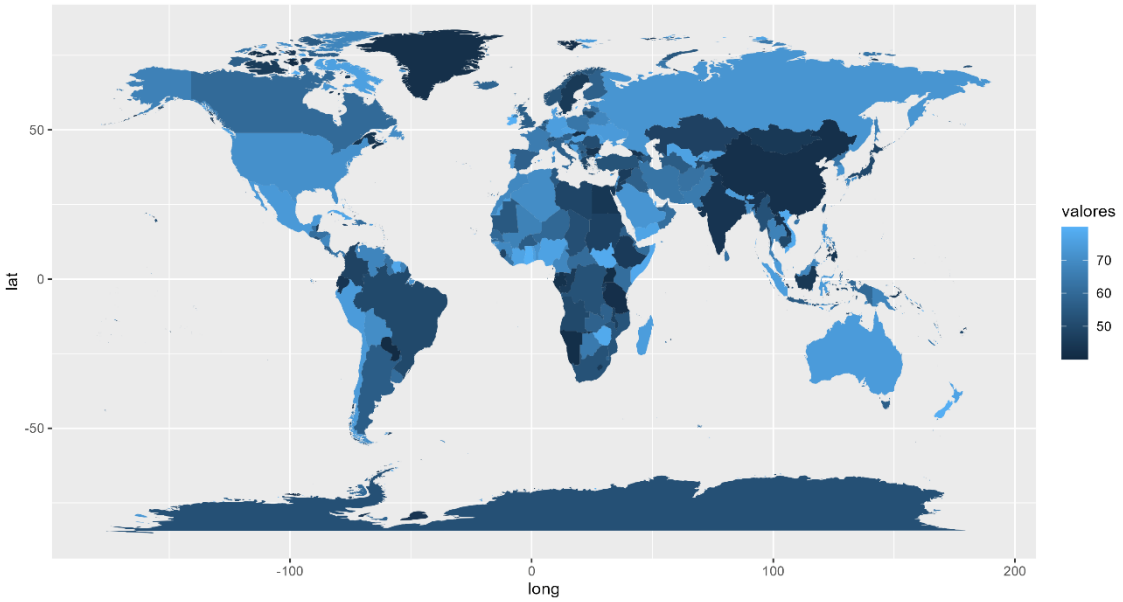


Figura 19.81. Países com valores de uma variável contínua diferentes.

Podemos ainda modificar a paleta de cores do gráfico. Existem diversas possibilidades, mas para trabalhar paletas em mapas, considero o método mais fácil o uso do pacote viridis. Com ele, adicionamos uma função de escala de preenchimento, que não é original do ggplot2, estabelecendo a paleta de cores que desejamos entre oito tipos. A função é scale_fill_viridis() e o parâmetro que devemos mudar para estabelecer a paleta é “option”.

Opções de paletas do pacote:

- "magma" (ou "A")
- "inferno" (ou "B")
- "plasma" (ou "C")
- "viridis" (ou "D")
- "cividis" (ou "E")
- "rocket" (ou "F")
- "mako" (ou "G")
- "turbo" (ou "H")

Veja como fica o mapa com nova paleta de cores do tipo plasma:

```
#Figura 19.82
#mudando paleta de cores do mapa
install.packages('viridis')
library(viridis)
```

```
ggplot(world) +
  geom_polygon(aes(x = long, y = lat, group = group, fill = valores), linewidth = 0)
+
  scale_fill_viridis(option = 'C')
```

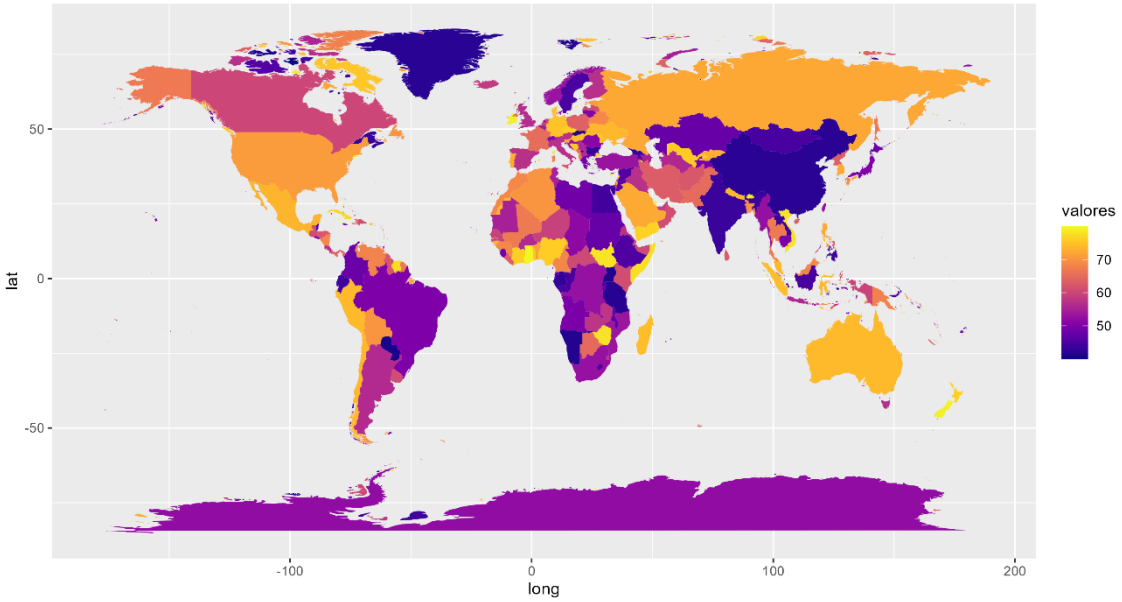


Figura 19.82. Atribuindo paleta de cores de preenchimento ao mapa.

Outra forma fácil de mexer em paletas ao construir mapas é utilizando a função scale_fill_distiller() do próprio ggplot2 (mas entenda que não é específica para paletas em mapas; você pode usar em outras situações). Particularmente gosto de trabalhar com elas quando desejo utilizar tons de cinza. Ela trabalha com alguns parâmetros que são interessantes:

- palette: especificação da paleta de cores. Podemos pôr um número ou código específico da paleta. Para os tons de cinza utilizo "Greys". Na documentação você encontrará os outros códigos possíveis.
- limits: estabelece os limites dos valores da legenda.
- direction: há paletas de cores que vai do branco (valores mais baixos) ao cinza escuro (valores mais altos), por exemplo. Caso queiramos inverter isso, podemos atribuir 1 ou -1 para alterar a direção das escalas de cores.
- name: alterar o título da legenda. Já vimos que o título da legenda pode ser alterado dentro da função labs(), nos parâmetros "colors" ou "fill". Mas há pessoas que ainda se confundem nisso, pois o nome "legenda" não está intuitivo na estética do ggplot.

```
#Figura 19.83
ggplot(world) +
```

```
  geom_polygon(aes(x = long, y = lat, group = group, fill = valores), linewidth = 0)
+
  scale_fill_distiller(palette = "Greys",
                       limits = c(40,80),
                       direction = 1,
                       name = "Título Aqui!")
```

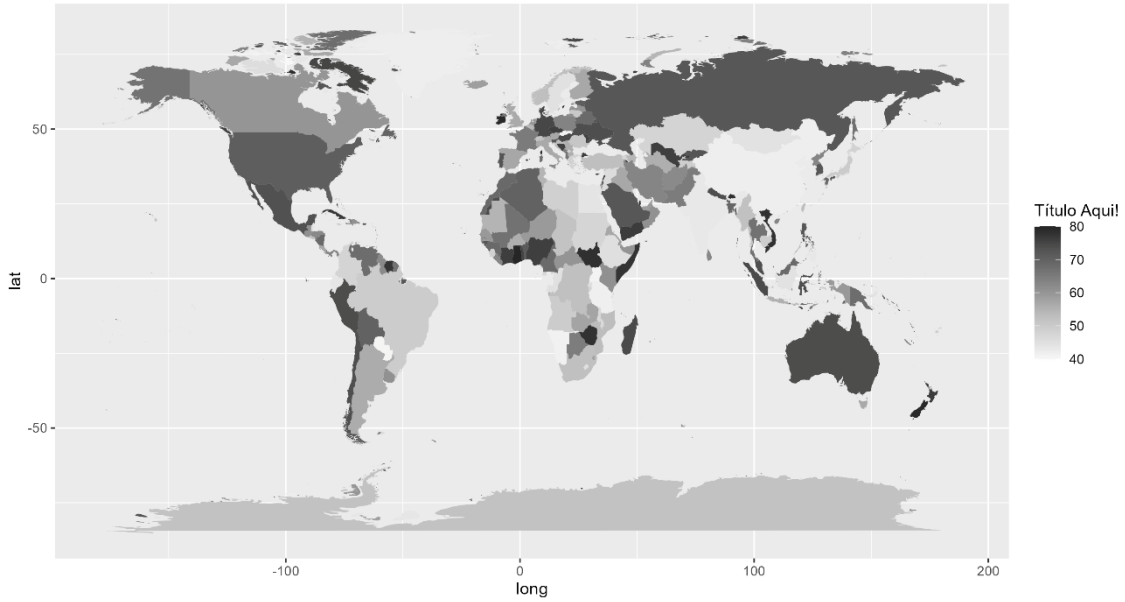


Figura 19.83. Paleta de cores em tons de cinza no mapa.

Até o momento trabalhamos com mapas a nível mundial. Para trabalharmos com mapas a nível nacional, precisamos utilizar as facilidades de um outro pacote chamado geobr. Tal pacote é mantido e atualizado pelo Instituto de Pesquisa Econômica Aplicada (IPEA). Com esse pacote podemos destrinchar melhor o espaço cartográfico brasileiro, explorar os estados e todos os municípios do Brasil. Por isso, ele será muito útil.

Alguns dados censitários estão contidos nos bancos do geobr e a fonte vem do IBGE. Em exemplos usaremos dados estaduais e depois municipais. Para cada nível de espaço geográfico há uma função específica a se utilizar, respectivamente: `read_state()` e `read_municipality()`. Entretanto, existe ricas possibilidades de dados. Podemos baixar dados de mesorregiões, microrregiões, vias de metrô, áreas urbanas, biomas, etc. Acesse https://ipeagit.github.io/geobr para ter acesso às demais funções e seus respectivos dados para construir seus mapas.

Primeiramente, vamos instalar e abrir a biblioteca geobr. Depois abrir os dados dos estados brasileiros e o Distrito Federal. A função exigirá que estabelecemos o ano da tomada de dados. Coloquei no exemplo 2020, pois é o último ano disponível até o momento. Se você não colocar o

ano, o R gerará um erro e nele mostrará os anos disponíveis. O site citado acima também mostrará os anos disponíveis para cada função que você pretende usar.

```
install.packages('geobr')
library(geobr)

#Informações por estado
dt_estados <- read_state(year = 2020)
view(dt_estados)
```

Ao abrir o banco de dados, você já verá algumas peculiaridades desse tipo de banco, que vem do IBGE. Há código numéricos para estados, regiões e municípios. Nesse presente banco, existe uma variável chamada "code_state" que traz os códigos de cada estado listado. Em "abbrev_state" estão as abreviações de cada estado, com duas letras. Em "name_state" estão os nomes por extenso de cada estado. Em "code_region" está o código para a região a qual o estado pertence (perceba que a região norte recebe o código 1). Em "name_region" temos o nome da região a qual o estado pertence. Em "geom" está a arquitetura dos polígonos de cada estado (nunca altera essa variável).

Tendo essa base, é possível então acrescentar qualquer tipo de informação sobre cada estado e depois disso confeccionarmos o nosso mapa a nível estadual. Para isso, vamos criar um banco extra com as informações de casos de dengue no Brasil em 2018. As informações foram tiradas do site G1 com a reportagem "Casos de dengue aumentam sete vezes no Brasil em 2019", acessível em https://g1.globo.com/bemestar/noticia/2019/09/11/casos-de-dengue-aumentam-sete-vezes-no-brasil-em-2019.ghtml. Após construir o banco chamado "dados_dengue", vamos juntar esse quantitativo de casos de dengue ao banco com os dados dos polígonos "dt_estados".

```
dados_dengue <- data.frame(
  abrev = c("RO","AC","AM","RR","PA","AP","TO","MA","PI","CE",
            "RN","PB","PE","AL","SE","BA","MG","ES","RJ","SP",
            "PR","SC","RS","MS","MT","GO","DF"),
  dengue = c(419,2447,1962,46,3332,608,1635,1869,1641,3645,
             20476,9717,9135,1617,168,7656,23290,7175,13176,11465,
             1070,140,93,2095,6178,73131,1605)
  )

#ou importe o banco
library(rio)
dados_dengue <- import('caminho_da_pasta/dados_dengue.txt')

#juntando os dados cartográficos com os de dengue
dados <- left_join(dt_estados, dados_dengue,
                   by = c('abbrev_state'='abrev'))
```

Como pode perceber, "dados_dengue" foi criado com as abreviações dos estados. Isso é importante para que haja compatibilidade entre os bancos e a junção de dados seja feita corretamente. Por fim, temos um novo banco de nome "dados".

Construindo o mapa, vamos utilizar dessa vez a função geom_sf(), que faz referência a *shapefile* (sf). Não precisará atribuir muita informação nela porque o geobr já fez isso para nós. Vamos apenas estabelecer preenchimento (*fill*) para a variável que tem os quantitativos de dengue em cada estado (variável chamada "dengue"). Vamos pôr "color" como NA (dado faltante) para que o mapa não mostre a linha de borda dos estados (perceberá que é outra forma de fazer isso, pois já fizemos isso usando linewidth = 0).

```
#Figura 19.84
#Casos de dengue no Brasil em 2018
ggplot(dados) +
  geom_sf(aes(fill = dengue), color = NA) +
  scale_fill_distiller(palette = "Reds",
                       direction = 1,
                       name = "Quant. casos")
```

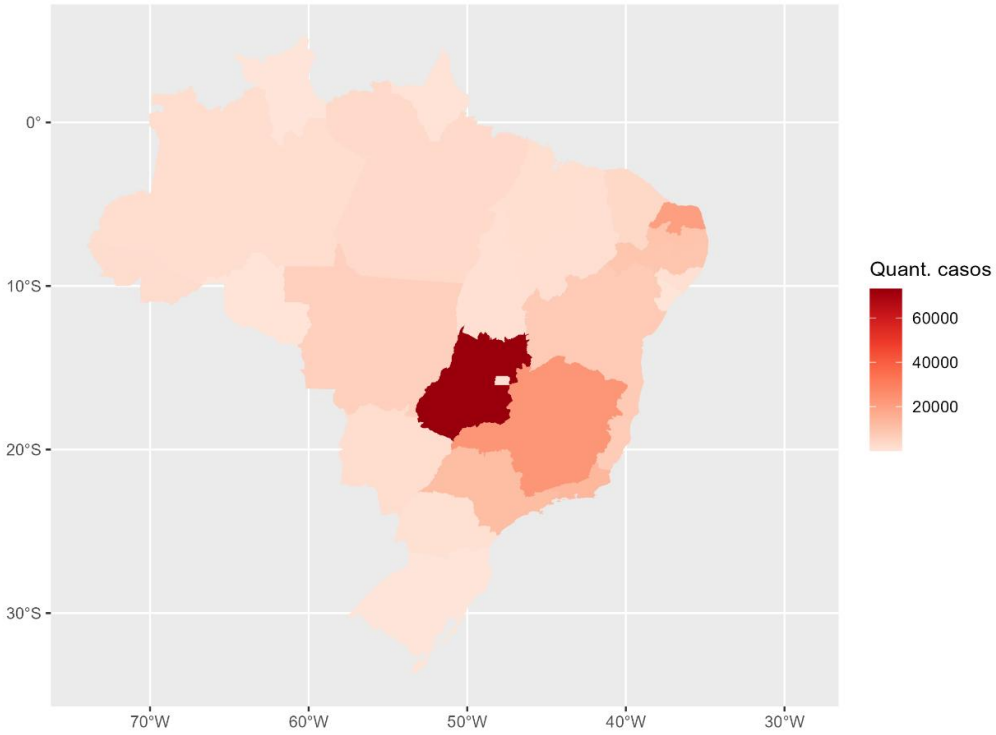


Figura 19.84. Mapa dos estados brasileiros em paleta vermelha de gradação.

No próximo exemplo, trabalharemos a nível municipal. Só que primeiro, filtraremos os municípios do estado do Piauí. Depois, vamos adicionar dados fictícios aleatoriamente, apenas para demonstrar a manipulação de dados contínuos municipais, usando runif(). O banco final se

chamará "dt_muni_PI". Para a nossa discussão, vamos supor que sejam os quantitativos de acidentes automobilísticos em um determinado ano. Veja como fica a modelagem dos dados e a criação do mapa.

```
#Figura 19.85
#Baixando municípios do Brasil no ano de 2020
dt_municipios <- read_municipality(year = 2020)

#Filtrando o estado do Piauí
dt_munic_PI <- dt_municipios %>%
  filter(abbrev_state == 'PI')

#Criando dados fictícios de acidentes de trânsito
set.seed(2222)
dt_munic_PI$acidentes <- runif(nrow(dt_munic_PI), min = 4, max = 100)

#Montando o mapa do Piauí com seus municípios
mapa_PI <- dt_munic_PI %>%
  ggplot() +
  geom_sf(aes(fill = acidentes), color = 'black') +
  scale_fill_distiller(palette = "Reds",
                       direction = 1,
                       name = "Quant. casos")

mapa_PI
```

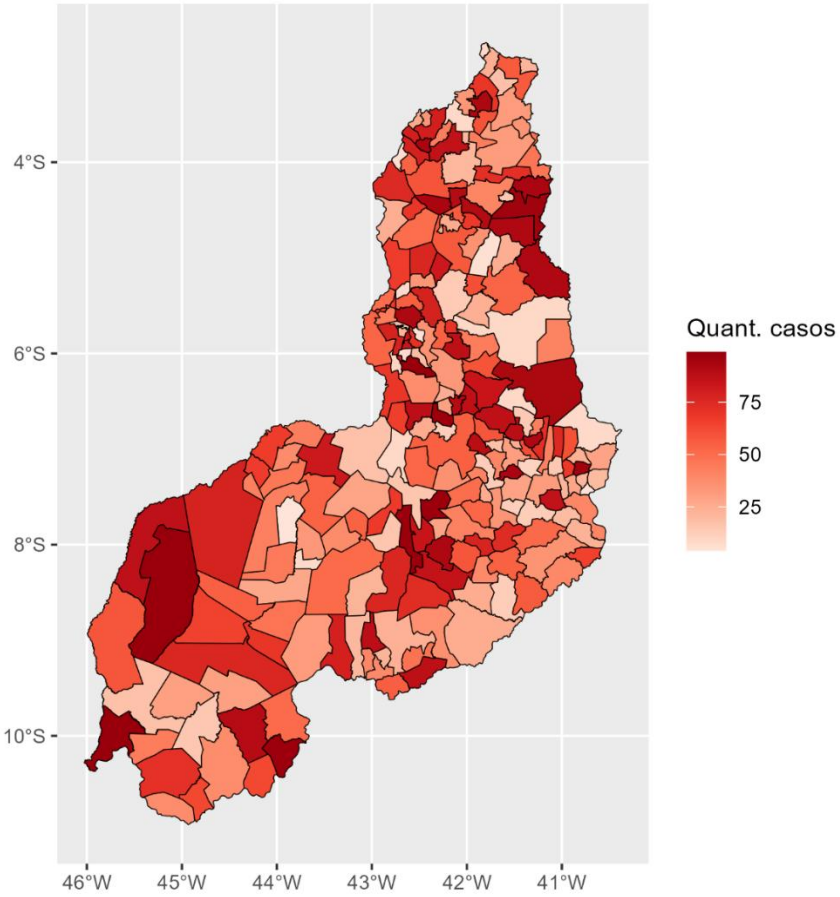


Figura 19.85. Mapa do Piauí em paleta vermelha por município.

19.2.25 Como salvar suas figuras

Para salvar qualquer gráfico ou figura vinda do ggplot2, podemos utilizar a função ggsave() do próprio pacote ggplot2. Nela há parâmetros importantes que devemos alterar, dependendo da exigência. São eles:

- filename = escrever o nome e a extensão do arquivo que deseja salvar, que pode ser em: eps, ps, tex (pictex), pdf, jpeg, tiff, png, bmp, svg ou wmf.
- plot = nome do objeto (figura) que deseja ser salvar. Se for omitido, será salvo por padrão o gráfico que estiver visível na aba Plots.
- units = unidade de medida do gráfico, caso você queira estabelecer tamanhos de comprimento e largura específicos. Pode ser em "in" (polegadas - padrão), "cm" (centímetros), "mm" (milímetros) ou "px" (pixels).
- width = numérico, largura da figura.
- heigth = numérico, altura da figura.
- dpi = resolução da imagem. Algumas revistas exigem uma resolução mínima de 300 dpi. Esse é o padrão do ggsave(). Se você deseja uma figura maior e mais nítida, coloque valor maior que 300, mas entenda que figura maior terá consequentemente maior o tamanho de arquivo. Sugiro não exagerar aqui.
- limitsize = booleano. O ggsave() não salva figuras maiores do que 50 x 50 polegadas por padrão. Para alterar esse limite (*limit size*), altere esse parâmetro para FALSE.

Sobre as extensões de arquivos, sugiro salvar em png, jpeg ou tiff. Frequentemente as imagens salvas como tiff (ou tif) gerarão arquivos maiores do que as duas extensões anteriores, mas a qualidade geralmente é superior. Algumas revistas científicas gostam da extensão svg. Para usar essa extensão certifique-se que o pacote svglite esteja instalado e abra a sua biblioteca antes de rodar a função ggsave().

Use a função getwd() para verificar qual a pasta o R está usando como diretório de trabalho. Sua figura será salva nessa pasta, a princípio. Caso queria salvá-la em outra pasta, basta especificar o caminho no momento de pôr o nome do arquivo.

Vamos então salvar o mapa do capítulo anterior, salvo no objeto chamado mapa_PI.

```
#A figura será salva nesta pasta:
getwd()
[1] "C:/Users/lucianogama/Documents"
```

```
ggsave('mapa_PI.png', mapa_PI)
Saving 4.21 x 3.53 in image
```

Logo depois de rodarmos o código com a função ggsave(), o R retorna o tamanho da figura salva: 4,21 por 3,53 polegadas.

Com o código abaixo, alteramos a unidade de medida para centímetros, o mapa ficou quadrado com extensões 50 x 50 cm e resolução de 500 dpi.

```
ggsave('mapa_PI.png', mapa_PI,
       units = 'cm', width = 50, height = 50,
       dpi = 500)
```

No momento de salvar os gráficos é crucial a conferência final. Abra o arquivo salvo e verifique sempre o tamanho dos textos. É comum a necessidade de aumentar as letras das escalas dos eixos e dos títulos.

Caso o gráfico construído não seja do ggplot, mas sim do graphics (pacote nativo do R), podemos utilizar a seguinte arquitetura:

```
Função para salvar o gráfico
O código do gráfico
dev.off()
```

Dependendo da extensão que você deseja salvar seu gráfico a função terá nome parecido. Temos: tiff(), png(), jpeg(), svg() e pdf(). Os parâmetros de maior importância que possamos configurar são: “width”, muda a largura do gráfico; “height”, muda a altura; “units”, altera a unidade de medida (pixels = “px” [padrão], centímetros = “cm”, milímetros = “mm” e polegadas = “in”); “res”, resolução em pdi; “quality”, qualidade de compressão da imagem (para melhor qualidade, use 100, que equivale a 100%).

Já a função dev.off() é necessária para finalizar todo o processo. Ele interrompe o controle sobre os dispositivos gráficos e o R entende que seu gráfico está finalizado para salvamento.

No exemplo a seguir, vamos salvar um simples gráfico de pontos em jpeg, com largura e altura de 50 centímetros, resolução de 300 dpi e qualidade de 100%. Veja o código para salvar a Figura 19.86:

```
jpeg('pontos.jpeg',
     width = 20,
     height = 20,
```

```
     units = 'cm',
     res = 300,
     quality = 100)
plot(x = seq(1,10,1), y = seq(1,10,1))
dev.off()
```

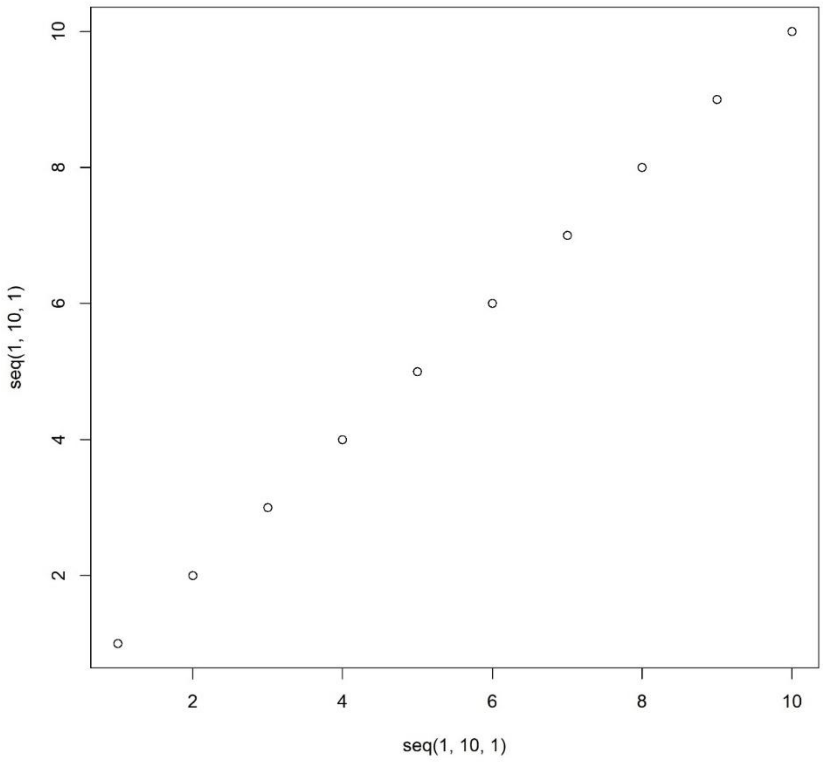


Figura 19.86. Gráfico de pontos gerado pelo graphics do R.

Capítulo 20

Referências

R Core Team (2023): A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. Disponível em: https://cran.r-project.org. Acesso em: 14/04/2024.

RSTUDIO DESKTOP. Posit Connect. Disponível em: https://posit.co/download/rstudio-desktop/. Acesso em: 14/03/2024.

TINN-R Editor. Disponível em: https://tinn-r.org/pt/. Acesso em: 14/03/2024.

BIO7. Disponível em: https://bio7.org/. Acesso em: 14/03/2024.

RKWARD 0.7.5. Disponível em: https://rkward.kde.org/. Acesso em: 14/03/2024.

MÜLLER, Kirill; WICKHAM, Hadley; FRANCOIS, Romain; BRYAN, Jennifer. Tibble package: Simple Data Frames, R package version 3.2.1, Disponível em: https://CRAN.R-project.org/package=tibble.

WICKHAM, H. et al. ggplot2 package: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016. Disponível em: https://ggplot2.tidyverse.org.

OPENAI. ChatGPT. Disponível em: https://chat.openai.com/. Acesso em: 14/03/2024.

WRITESONIC. Chatsonic. Disponível em: https://writesonic.com/chat. Acesso em: 14/03/2024.

GEMINI. Chat to supercharge your ideas. Disponível em: https://gemini.google.com/app. Acesso em: 14/03/204.

NOTEPAD++ 8.6.6. Disponível em: https://notepad-plus-plus.org/downloads/. Acesso em: 14/03/2024.

BRASIL. Lei n°13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais – LGPD. Brasília, DF: Diário Oficial da União, 2018.

WICKHAM, H. et al. readr package: Read Rectangular Text Data. R package version 2.1.4, https://CRAN.R-project.org/package=readr.

WICKHAM, H, et al. Welcome to the tidyverse. JOSS: The Joural of Open Source Sorftware, 4(43), p 1686, 2019. doi:10.21105/joss.01686.

BUUREN, Stef van; GROOTHUIS-OUDSHOORN, Karin. mice: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software, 45(3), p 1-67, 2011. doi: 10.18637/jss.v045.i03.

BACHE, Stefan Milton; WICKHAM, Hadley; HENRY, Lionel. Magrittr package: A Forward-Pipe Operator for R. R package version 2.0.3, Disponível em: https://CRAN.R-project.org/package=magrittr.

FISHER, R.A. The use of multiple measurements in taxonomic problems. Annals of Eugenics. V7(2), p 179-188, 1936. Doi: https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

KAPLAN, Jacob. fastDummies package: Fast Creation of Dummy (Binary) Columns and Rows from Categorical Variables. R package version 1.7.3. Disponível em: https://CRAN.R-project.org/package=fastDummies.

BALLINGS, Michel; POEL, Dirk Van den. Dummy package: Automatic Creation of Dummies with Support for Predictive Modeling. R package version 0.1.3. Disponível em: https://CRAN.R-project.org/package=dummy.

ARA, Tohiaki. Makedummies package: Create Dummy Variables from Categorical Data. R package version 1.2.1. Disponível em: https://CRAN.R-project.org/package=makedummies.

Github. Pasta - lucianogama/BioEpiR. Disponível em https://github.com/lucianogama/BioEpiR/. Acesso em: 11/05/2024.

TRIOLA, Mario F. Introdução a Estatística. 12ª ed, Rio de Janeiro: LTC, 2017.

VIEIRA, Sonia. Introdução à Bioestatística. 4ª ed, São Paulo: Elsevier, 2011.

WICKHAM, H, et al. dplyr package: A Grammar of Data Manipulation. R package version 1.1.2. Disponível em: https://CRAN.R-project.org/package=dplyr.

SCHEFFER, Judi. Dealing with Missing Data. Res Lett Inf Math Sci, v. 3, p 153-160, 2002.

POTVIN, C.; LECHOWICZ, M. J.; TARDIF, S. The statistical analysis of ecophysiological response curves obtained from experiments involving repeated measures, Ecology, v. 71, 1389-1400, 1990.

McNeil, Donald. R. Interactive Data Analysis: A practical primer. New York: Wiley, 1977.

ZAGO, MA, FALCÃO, RP, PASQUINI, R. Tratado de Hematologia. 1ª ed. Rio de Janeiro: Atheneu, 2013. 823 p.

WUERTZ, D. et al. fBasics package: Rmetrics - Markets and Basic Statistics. R package version 4022.94. Disponível em: https://CRAN.R-project.org/package=fBasics.

SCHOBER, P; BOER, Christa; SCHWARTE, Lothar A. Correlation coefficients: Appropriate use and interpretation. Anesthesia and Analgesia, 126(5), 1763-1768, 2018. doi: 10.1213/ANE.0000000000002864

BECKER, R. A.; CHAMBERS, J. M.; WILKS, A. R. The New S Language: A Programming Environment for Data Analysis and Graphics. Wadsworth & Brooks/Cole, 1988.

HAMNER, Ben; FRASCO, Michael; LEDELL, Erin. Metrics package: Evaluation Metrics for Machine Learning. R package version 0.1.4. Disponível em: https://CRAN.R-project.org/package=Metrics.

JAMES, G et al. An Introduction to Statistical Learning: With Applications in R. 7ª ed, Springer; 2013.

MENARD, Scott. Applied Logistic Regression Analysis. 2ª ed, SAGE Publications, Inc; 2001.

JOVIĆ, A; BRKIĆ, K.; BOGUNOVIĆ, N. A review of feature selection methods with applications. MIPRO, 2015, 25-29.

FOX, J et al. car package: An R Companion to Applied Regression. Disponível em: https://cran.r-project.org/web/packages/car.

APOVIAN, Caroline M. Obesity: definition, comorbidities, causes, and burden. Am J Manag Care, v. 22(7):s176-85, 2016. PMID: 27356115.

BECKER, J. et al. rio package: A Swiss-army knife for data file I/O. R package version 0.5.29. Disponível em: https://cran.r-project.org/web/packages/rio.

MACKAY, T.F.C; ANHOLT, R.R.H. Gregor Mendel's legacy in quantitative genetics. PLoS Biol, v. 20(7), 2022. doi: 10.1371/journal.pbio.3001692. PMID: 35852997.

AGRESTI, Alan. An Introduction to Categorical Data Analysis, 2ª ed. New York: John Wiley & Sons, 2007.

GREENBLATT, D.J. et al Gender differences in pharmacokinetics and pharmacodynamics of zolpidem following sublingual administration. J Clin Pharmacol, v. 54(3):282-90, 2014. doi: 10.1002/jcph.220.

CAPP, Edison; NIENOV, Otto Henrique. Bioestatística quantitativa aplicada. Universidade Federaral do Rio Grande do Sul: Faculdade de Medicina. Porto Alegre, 2020.

PARAJULI, Subash. Quantitative methods using R. Disponível em: https://bookdown.org/subashparajuli/quant-r/one-way-anova.html. Acesso em: 11/07/2023

SIGNORELL A. et al. DescTools package: Tools for Descriptive Statistics. R package version 0.99.48. Disponível em: https://CRAN.R-project.org/package=DescTools.

LAWRENCE, M.A. ez package: Easy Analysis and Visualization of Factorial Experiments. R package version 4.4-0. Disponível em: https://CRAN.R-project.org/package=ez.

KASSAMBARA, Alboukadel. rstatix package: Pipe-Friendly Framework for Basic Statistical Tests. R package version 0.7.2. Disponível em: https://CRAN.R-project.org/package=rstatix.

POHLERT, Thorsten. PMCMRplus: Calculate Pairwise Multiple Comparisons of Mean Rank Sums Extended. R package version 1.9.6. Disponível em: https://CRAN.R-project.org/package=PMCMRplus.

COHEN, Jacob. Coefficient of agreement for nominal scales. Journal of Educational and Measurement, New York, v.20(1), p.37-46, 1960. doi: 10.1177/001316446002000104

MCHUGH, Mary L. Interrater reliability: the kappa statistic. Lessons in biostatistics. Biochemia Medica, Zagreb, v.22(3): 276-82, 2012.

CARDOSO, E. S. de C. et al. The use of saliva as a practical and feasible alternative to urine in large-scale screening for congenital cytomegalovirus infection increasesinclusion and detection rates. Revista Da Sociedade Brasileira De Medicina Tropical, v.48(2), 206-207, 2015. doi: https://doi.org/10.1590/0037-8682-0200-2014

GAMER, M.; LEMON, J; SINGTH I.F.P. irr package: Various Coefficients of Interrater Reliability and Agreement. R package version 0.84.1. Disponível em: https://CRAN.R-project.org/package=irr.

HIRAKATA, Vânia Naomi; CAMEY, Suzi Alves. Análise de concordância entre métodos de Bland-Altman. Ver. HCPA & Fac. Med. Univ. Fed. Rio Gd. do Sul, v.29(3): 261-268, 2009.

ALTMAN, D.G.; BLAND, J.M. Measurement in Medicine: The analysis of method comparison studies. Journal of the Royal Statistical Society. Series D (The Statistician), v. 32(3):307-317, 1983.

DATTA, D. blandr package: a Bland-Altman Method Comparison package for R. Disponível em: https://github.com/deepankardatta/blandr.

IBGE. Instituto Brasileiro de Geografia e Estatística: Censo 2022. Disponível em: https://www.ibge.gov.br/. Acesso em: 20/04/2024.

COHEN, Jacob. Statistical power analysis for the behavioral scence. 2ª ed. New York: Lawrence Erlbaum Associates, Department of Psychology, New York University, New York, 1988.

CHAMPELY, S. et al. pwr package: Basic Functions for Power Analysis. R package version 1.3-0. Disponível em: https://CRAN.R-project.org/package=pwr.

ZHANG, Z et al. WebPower package: Basic and Advanced Statistical Power Analysis. R package version 0.9.4. Disponível em: https://CRAN.R-project.org/package=WebPower.

WebPower. Statistical power analysis online. Disponível em: https://webpower.psychstat.org/wiki/models/index. Acesso em: 20/04/2024.

DEMIDENKO, Eugene. Sample size determination for logistic regression revisited. Statist. Med, v. 26:3385-3397, 2006. doi: 10.1002/sim.2771

SILVA, R. C. F. et al. PPGCS Sample Size [Computer software]. Disponível em: www.oraculoestatistico.com.br/ppgcssamplesize/public. Acesso em: 15/01/2024

BOLFARINE, Heleno; BUSSAB, Wilton O. Elementos de amostragem. São Paulo: Editora Blucher, 2005.

KUHN, M. et al. caret package: Classification and Regression Training. Disponível em: https://cran.r-project.org/web/packages/caret.

YAN, Y achen. MLmetrics package: Machine Learning Evaluation Metrics. R package version 1.1.1. Disponível em: https://CRAN.R-project.org/package=MLmetrics.

OBUCHOWSKI, Nancy A,; BULLEN, Jennifer A. Receiver operating characteristic (ROC) curves: review of methods with applications in diagnostic medicine. Phys Med Biol, v. 29;63(7):07TR01, 2018. doi: 10.1088/1361-6560/aab4b1. PMID: 29512515.

BRAGA, Ana Cristina da Silva. Curvas ROC: aspectos funcionais e aplicações. 267 f. Dissertação - Universidade do Minho, Braga. 2000.

AKOBENG, Anthony K. Understanding diagnostic tests 3: Receiver operating characteristic curves. Acta paediatrica, Oslo, v. 96(5):644–7, 2007. doi: 10.1111/j.1651-2227.2006.00178.x.

YOUDEN, W. J. Index for rating diagnostic tests. Cancer, v. 3(1):32–5, 1950. doi: 10.1002/1097-0142.

LOPEZ-RATON, Monica; RODRIGUEZ-ALVAREZ, Maria Xose. OptimalCutpoints package: An R Package for Selecting Optimal Cutpoints in Diagnostic Tests. Disponível em: https://cran.r-project.org/web/packages/OptimalCutpoints.

AMARO, A. et al. Plasma leukocyte elastase concentration in angiographically diagnosed coronary artery disease. European Heart Journal, v.16(5), 615-622, 1995. doi: 10.1093/oxfordjournals.eurheartj.a060964.

SOPELETE, Mônica C. Métodos de análise em estudos sobre diagnóstico. In: MINEO, J.R.; SILVA, D.A.O.; SOPELETE, M.C.; LEAL, G.S.; VIDIGAL, L.H.G.; TÁPIA, L.E.R.; BACCHIN, M.I. Pesquisa na área biomédica: do planejamento à publicação. Uberlândia: EDUFU p.203-223, 2005. ISBN: 978-85-7078-523-7. https://doi.org/10.7476/9788570785237.0009.

PEREIRA, Maurício Gomes. Epidemiologia: teoria e prática. 6. ed. Rio de Janeiro: Guanabara Koogan, 2002.

DICIO (Dicionário Online de Português). Surto. Disponível em: https://www.dicio.com.br/surto. Acesso em: 23/04/2024.

DICIO (Dicionário Online de Português). Epidemiologia. Disponível em: https://www.dicio.com.br/epidemiologia/. Acesso em: 23/04/2024.

DICIO (Dicionário Online de Português). Evolução. Disponível em: https://www.dicio.com.br/evolucao. Acesso em: 23/04/2024.

SPURIOUS CORRELATIONS: correlation is not causation. Disponível em: https://www.tylervigen.com/spurious-correlations. Acesso em: 30/04/2024.

FLETCHER, Robert H.; FLETCHER, Suzanne W. Epidemiologia clínica: elementos essenciais. 4a ed. Artmed, 2008.

DEMARCHI, Izabel Galhardo; CHATALOV, Renata Cristina Souza. Bioestatística e epidemiologia. Maringá - PR: Unicesumar, 2021, 328 p.

SILVA, V. R. et al. Tendência temporal das taxas de incidência e de mortalidade por COVID-19 e sua relação com indicadores socioeconômicos no Piauí: estudo ecológico, 2020-2021. Epidemiol. Serv. Saúde, v. 31(2):202216, 2022. doi 10.1590/S2237-96222022000200022.

KIND, Luciana; CORDEIRO, Rosineide. Narrativas sobre a morte: a gripe espanhola e a covid-19 no Brasil. Psicol. Soc., v. 32, 2020. doi: https://doi.org/10.1590/1807-0310/2020v32240740.

MERCHÁN-HAMANN, Edgar. Diagnóstico macrorregional da situação das endemias das Regiões Norte e Nordeste. Inf. Epidemiol. Sus, Brasília, v. 6(3), p. 43-114, 1997.

RENTE, Arthur; JUNIOR, Delcio, U.; UEZATO, Karina M. K. Coronavírus e o Coração: Um Relato de Caso sobre a Evolução da COVID-19 Associado à Evolução Cardiológica. Arq. Bras. Cardiol., v. 114(5), 2020. doi: https://doi.org/10.36660/abc.20200263.

ROY-GARCÍA, I.A. et al. ROC curves: general characteristics and their usefulness in clinical practice. Rev Med Inst Mex Seguro Soc., v. 61(3):S497-502, 2023. doi: 10.5281/zenodo.8319791.

LIMA, L. V. et al. Temporal trend, distribution and spatial autocorrelation of leprosy in Brazil: ecological study, 2011 to 2021. Revista Brasileira de Epidemiologia, v. 25, 2022. doi: 10.1590/1980-549720220040.

CAZOLARI, P. G. et al. Burnout and well-being levels of medical students: a cross-sectional study. Revista Brasileira De Educação Médica, 44(4), 2020. doi: 10.1590/1981-5271v44.4-20190138.

OKUYAMA, J. H. H.; GALVÃO, T. F.; SILVA, M. T. Intoxicações e fatores associados ao óbito por agrotóxicos: estudo caso controle, Brasil, 2017. Revista Brasileira De Epidemiologia, v. 23, 2020. doi: 10.1590/1980-549720200024.

CAILLEAUX-CEZAR, M. et al. Impact of smoking on sputum culture conversion and pulmonary tuberculosis treatment outcomes in Brazil: a retrospective cohort study. Jornal Brasileiro De Pneumologia, v. 44(2), 99–105, 2018. doi: 10.1590/S1806-37562017000000161.

BAVARESCO, T.; LUCENA, A. F. Low-laser light therapy in venous ulcer healing: a randomized clinical trial. Revista Brasileira De Enfermagem, v. 75(3), 2022. doi: 10.1590/0034-7167-2021-0396

BARUFALDI, L. A. et al. Risco de iniciação ao tabagismo com o uso de cigarros eletrônicos: revisão sistemática e meta-análise. Ciência & Saúde Coletiva, 26(12), 6089–6103, 2021. doi: 10.1590/1413-812320212612.35032020

STEVENSON, M. et al. epiR package: Tools for the Analysis of Epidemiological Data. R package version 2.0.66. Disponível em: https://CRAN.R-project.org/package=epiR.

SJOBERG, D. D. et al. gtssumary package: Presentation-Ready Data Summary and Analytic Result Tables. Disponível em: https://cran.r-project.org/web/packages/gtsummary.

GOHEL, D. et al. flextable package: Functions for Tabular Reporting. R package version 0.9.4. Disponível em: https://CRAN.R-project.org/package=flextable.

GRAPHICS. The R graphics version 4.5.0 package documentation. Disponível em: https://stat.ethz.ch/R-manual/R-devel/library/graphics/html/00Index.html. Acesso em: 02/04/2024.

CHAMBERS, J. M. et al. Graphical Methods for Data Analysis. Belmont, CA: Wadsworth, 1983.

GGPLOT2. Aesthetic specifications. Disponível em: https://tidyverse.github.io/ggplot2-docs/articles/ggplot2-specs.html. Acesso em: 07/02/2024.

CRAMPTON, E. W. The growth of the odontoblast of the incisor teeth as a criterion of vitamin C intake of the guinea pig. The Journal of Nutrition, 33(5), 491–504, 1947. doi: 10.1093/jn/33.5.491.

COLOR-HEX. Color hex color codes. Disponível em: www.color-hex.com. Acesso em: 07/02/2024.

GUIDONI, Karlo. Paletas de cores disponíveis no R. Disponível em: https://karloguidoni.com/post/2020-07-26-paletas-de-cores-disponiveis-no-r/. Acesso em: 09/02/2024.

GGTHEMES. All your figure are belong to us. Disponível em: https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/. Acesso em: 09/02/2024

GGPLOT2. Function reference. Disponível em: https://ggplot2.tidyverse.org/reference/. Acesso em: 09/02/2024.

PEDERSEN, T. L. patchwork package: The Composer of Plots. R package version 1.1.2. Disponível em: https://CRAN.R-project.org/package=patchwork.

YU, Guanghuang. Ggplotify package: Convert Plot to 'grob' or 'ggplot' Object. R package version 0.1.2. Disponível em: https://CRAN.R-project.org/package=ggplotify.

GARNIER, S. et al. viridis package: Colorblind-Friendly Color Maps for R. R package version 0.6.3. Disponível em: https://cran.r-project.org/web/packages/viridis.

PEREIRA, R. et al. geobr package: Download Official Spatial Data Sets of Brazil. R package version 1.8.2. Disponível em: https://CRAN.R-project.org/package=geobr.

G1. Casos de dengue aumentam sete vezes no Brasil em 2019. Disponível em: https://g1.globo.com/bemestar/noticia/2019/09/11/casos-de-dengue-aumentam-sete-vezes-no-brasil-em-2019.ghtml. Acesso em: 14/03/02024.

COGHLAN, Avril. Biomedical Statistics 0.2 documentation: Little book of R for biomedical statistics. Disponível em: https://a-little-book-of-r-for-biomedical-statistics.readthedocs.io/en/latest/. Acesso: em: 14/03/2024.

MIOT, Hélio Amarante. Tamanho da amostra em estudos clínicos e experimentais. J Vasc Bras, v. 10(4), 2011.

FOX, John. Applied regression analysis and generalized linear models. 3ª ed. London: McMaster University. 2016.

LIMA, Roberval. Distribuição Normal-site. Disponível em: https://rstudio-pubs-static.s3.amazonaws.com/191913_83dca87aee2648d6a7725b9e68cd107f.html. Acesso em: 15/03/2024.

SOUZA, Fernando. Principais IDE's para R. Disponível em: https://producaoanimalcomr.wordpress.com/2018/02/04/principais-ides-para-r/. Acesso em: 14/08/2023.

ANUNCIAÇÃO, Luis. Conceitos e análises estatísticas com R e JASP. Disponível em: https://bookdown.org/luisfca/docs/anova-de-medidas-repetidas.html. Aceso em: 16/08/2023.

FARAWAY, Julian J. Pratical regression and Anova using R. 2002. Disponível em: https://cran.r-project.org/doc/contrib/Faraway-PRA.pdf. Acesso em: 14/12/2023.

FLEISS, Joseph L. Measuring nominal scale agreement among many raters. Psychological Bulletin, v. 76(5), 378-38, 1971.

FLEISS, J.L.; LEVIN, B.; PAIK, M.C. Statistical methods for rates and proportions. 3ª ed. New York: John Wiley & Sons, 2003.

TUSTUMI, Francisco. Choosing the most appropriate cut-point for continuous variables. Rev Col Bras Cir, v. 49, 2022. doi: 10.1590/0100-6991e-20223346.

SIQUEIRA, Arminda Lucia; TIBÚRCIO, Jacqueline Domingues. Estatística na área da saúde. COOPMED; 1ª edição, 520 p, 2011.

UNAL, Ilker. Defining an optimal cut-point value in roc analysis: an alternative approach. Comput Math Methods Med, v. 2017, 2017. doi: 10.1155/2017/3762651.

RUOPP, M. D. et al. Youden index and optimal cut-point estimated from observations affected by a lower limit of detection. Biom J., v. 50(3):419-30, 2008. doi: 10.1002/bimj.200710415.

WICHHAM, H.; CETINKAYA-RUNDEL, M.; GROLEMUND, G. R for data Science – Import, tidy, transform, visualize, and model data. 2ª ed. O'Reilly. Disponível em: https://r4ds.hadley.nz/. Acesso em: 21/11/2023.

Capítulo

21

Listas

LISTA DE QUADROS

Quadro 1.1. Operadores matemáticos e lógicos do R.

Quadro 13.1. Passos para os cálculos da ANOVA.

Quadro 19.1. Tamanho médio de pápula para os principais alérgenos em uma população X.

Quadro 19.2. Argumentos que podem ser utilizados no parâmetro “statistic” da função tbl_summary.

LISTA DE TABELAS

Tabela 10.1. Abordagem interpretar um coeficiente de correlação.

Tabela 11.1 Resumo interpretativo de métricas na comparação de modelos.

Tabela 12.1. Associação entre os métodos de aprendizagem e o perfil de personalidade de alunos.

Tabela 12.2. Associação entre os métodos de aprendizagem e o perfil de personalidade de alunos.

Tabela 12.3. Associação entre os métodos de aprendizagem e o perfil de personalidade de alunos.

Tabela 13.1. Comparações das dosagens de CA15-3 sérico entre casos e controles.

Tabela 14.1 Nível de concordância segundo valor de kappa.

Tabela 15.1. Tomada de decisão de um teste de hipótese.

Tabela 15.2. Tabela de frequência entre diabéticos e suas dosagens categorizadas de glicose sérica em jejum.

Tabela 16.1. Tabela de contingência 2x2.

Tabela 16.2. Tabela de contingência 2x2 com valores.

Tabela 17.1. Relação da AUC e acurácias.

Tabela 18.1. Tabela de contingência 2x2 renomeada.

Tabela 18.2. Tabela de contingência para o uso de bebida alcoólica diariamente e pancreatite aguda.

Tabela 18.3. Tabela de contingência dos casos de câncer de estômago e o consumo de frutas regularmente.

Tabela 18.4. Tabela de contingência dos casos de infarto agudo do miocárdio e o consumo de 50 gramas de açúcar diariamente.

Tabela 18.5. Tabela de contingência modifica a partir da função table().

Tabela 18.6. Tabela de contingência para casos de miopia em relação ao uso de colírios à base de corticoides.

Tabela 18.7. Tabela de contingência para problemas dentários e ingestão diária de doces e achocolatados.

Tabela 18.8. Tabela de contingência para a relação entre diabetes mellitus do tipo II e práticas regulares de exercício físico.

Tabela 19.1. Tamanho médio de pápula para os principais alérgenos em uma população X.

Tabela 19.2. Resumo dos resultados de cada variável presente em trial.

Tabela 19.3. Modificação dos nomes das variáveis.

Tabela 19.4. Modificação dos nomes das classes das variáveis categóricas.

Tabela 19.5. Estabelecendo uma variável dependente (desfecho).

Tabela 19.6. Tratando dados ausentes.

Tabela 19.7. Trocando a mediana e o intervalo interquartílico pela média e desvio padrão para a idade; removendo dados faltantes.

Tabela 19.8. Padronizando a quantidade de números após a vírgula.

Tabela 19.9. Acrescentando funções acessórias.

LISTA DE FIGURAS

Figura 1.1. O ambiente R de programação.

Figura 1.2. O ambiente RStudio.

Figura 1.3. Janela do Editor.

Figura 1.4. Janela do Console.

Figura 1.5. Janela com as abas *Environment* e *History*.

Figura 1.6. Janela com as abas *Files*, *Plots*, *Packages*, *Help* e *Viewer*.

Figura 1.7. Aba Help e a busca de ajuda da função round().

Figura 2.1. Os quatro tipos de variáveis na estatística.

Figura 2.2. Banco de dados estruturado.

Figura 2.3. Janela do Editor de dados no RStudio.

Figura 2.4. Inserindo dados e modificando o nome e o tipo de variáveis.

Figura 2.5. Importação de dados.

Figura 2.6. Janela do "From Text (readr)".

Figura 5.1. Dados em uma distribuição normal.

Figura 5.2. Distribuição normal de dados com a média, mediana e moda definidas.

Figura 5.3. Distribuição fora da normal.

Figura 5.4. Distribuição fora da normal, com a média, mediana e moda definidas.

Figura 6.1. Esquema de montagem de um diagrama de caixa (*boxplot*).

Figura 6.2. Um único *boxplot*.

Figura 6.3. *Boxplots* de grupos diferentes.

Figura 7.1. *Boxplot* de peso por grupo.

Figura 8.1. Distribuição Z, suas zonas de valores críticos e conclusão sobre H_0.

Figura 8.2. Esquema com os valores tabelados e o calculado do teste t.

Figura 8.3. Diferentes níveis de significância e suas áreas de rejeição de H_0.

Figura 9.1. Histogramas representado distribuições quantitativamente diferentes.

Figura 9.2. Distribuição normal.

Figura 9.3. Simetria de distribuições de dados.

Figura 9.4. Comparação visual entre distribuições normal e não normal.

Figura 9.5. Q-Q plots das variáveis dados1 e dados2 do banco cap9.

Figura 10.1. Diagrama de dispersão entre IMC e pressão arterial sistólica.

Figura 10.2. Diagrama de dispersão entre IMC e o HDL sérico.

Figura 10.3. Diagrama de dispersão entre IMC e os níveis de testosterona total sérica.

Figura 10.4. Esquema para saber qual coeficiente de correlação utilizar.

Figura 11.1. Reta de regressão e seus resíduos

Figura 11.2. Histogramas demonstrando a diferença de espectros entre variáveis originais e modificadas pelo escore Z.

Figura 13.1 Fluxograma para o uso de testes de comparação de médias.

Figura 13.2. Esquema para realizar o teste t de *Studant* para amostras independentes.

Figura 13.3. Histograma dos níveis de CA15-3 entre controles e casos.

Figura 13.4. Formas diferentes de demonstrar graficamente resultados de comparação entre duas médias.

Figura 13.5. Comparação des médias par a par através do teste Tukey.

Figura 13.6. Comparação da média de glicose entre os grupos de estudo.

Figura 13.7. *Boxplot* mostrando a taxa de glicose entre grupos de diabéticos.

Figura 13.8. Distribuição das dosagens de IgG entre os tempos T0, T1, T2 e T3.

Figura 13.9. Comparação entre as dosagens médias de IgG raiva-específica antes e após vacina.

Figura 13.10. Distribuição das dosagens de IgG entre os tempos T0, T1, T2 e T3.

Figura 14.1. Gráfico de Bland-Altman para análise de concordância entre variáveis numéricas.

Figura 14.2. Padrões gráficos mostrando falta de concordância entre medições.

Figura 15.1. Diferença dimensional entre população e amostra

Figura 17.1. Modelo genérico de reconhecimento de Thrustone para representação de detecção sensorial.

Figura 17.2. Exemplo de uma curva ROC.

Figura 17.3. A área sob a curva ROC.

Figura 17.4. Diferença gráfica entre as áreas sob a curva quando iguais a 1,0 e 0,5, respectivamente.

Figura 17.5. Curvas ROC (A) e modelos de Thrustone (B).
Figura 17.6. Curva ROC do prognóstico da hemorragia subaracnoidea aneurismática segundo a S100B em ng/mL.

Figura 17.7. Curva ROC do prognóstico da hemorragia subaracnoidea aneurismática segundo a S100B, hachurada e com o valor de AUC.

Figura 17.8. Comparando duas curvas ROC.
Nota: Em preto a dosagem de S100B e em vermelho a idade.

Figura 17.9. Curva ROC para estabelecer o ponto de corte usando o método de Youden.

Figura 17.10. Gráfico de densidade das distribuições de elastase entre casos e controles.

Figura 18.1. Esquema representativo da incidência e prevalência
Figura 18.2. Tipos de estudos epidemiológicos.

Figura 19.1 Gráficos construídos a partir do pacote graphics do R.

Figura 19.2. Histograma via ggplot.

Figura 19.3. Histograma com seis barras.

Figura 19.4. Histograma com espectro diferente do padrão.

Figura 19.5. Histograma com cor modificada.

Figura 19.6. Histograma de populações diferentes.

Figura 19.7. Gráfico de densidade.

Figura 19.8. Gráfico de densidade preenchido e transparência de 0,5.

Figura 19.9. Gráfico de densidade de grupos diferentes.

Figura 19.10. Q-Q plot demonstrando uma distribuição normal.

Figura 19.11. Q-Q plot demonstrando uma distribuição fora da normal.

Figura 19.12. Q-Q plot com os resíduos de regressão linear.

Figura 19.13. Q-Q plot com os resíduos de regressão linear via ggplot.

Figura 19.14. Gráfico de dispersão ou gráfico de pontos.

Figura 19.15. Tipos de formas dos pontos possíveis.
Fonte: https://tidyverse.github.io/ggplot2-docs/articles/ggplot2-specs.html

Figura 19.16. Gráfico de dispersão com "pontos" de forma quadrada.

Figura 19.17. Gráficos de tendência *smooth* (linha azul).

Figura 19.18. Tipos de linhas possíveis no ggplot.

Figura 19.19. Linhas de tendência com métodos diferentes de tendência (linear e polinomial local).

Figura 19.20. Gráfico de linhas simples.

Figura 19.21. Gráfico de linhas com parâmetros de linha modificados.

Figura 19.22. Gráfico de área.

Figura 19.23. Gráfico de área com parâmetros de preenchimento modificados.

Figura 19.24. Gráfico de barras usando geom_bar().

Figura 19.25. Gráfico de barras com geom_col(), posto na horizontal pela quantidade de classes.

Figura 19.26. Gráfico de barras com classes ordenadas.

Figura 19.27. Diagrama de caixa ou *boxplot* simples.

Figura 19.28. *Boxplots* de classes diferentes.

Figura 19.29. *Boxplots* de classes diferentes preenchidos por grupos distintos.

Figura 19.30. Gráfico de pontos sobrepostos ou *jittered points.*

Figura 19.31. Diferença de gráfico de pontos por classe e *jittered points.*

Figura 19.32. Gráfico violino por grupos.

Figura 19.33. Gráfico violino com quartis.

Figura 19.34. Gráfico violino com médias.

Figura 19.35. Gráfico de barras com barras de erro.

Figura 19.36. Gráfico de barras com barras de erro esteticamente modificadas.

Figura 19.37. Gráfico de barras com caixas de erro.

Figura 19.38. Gráfico de barras com ponto e linhas de erro.

Figura 19.39. Gráfico de barras com barras de erro e valores de média.

Figura 19.40. Gráfico de barras com barras de erro e valores de média em forma de rótulo.

Figura 19.41. Gráfico com textos soltos em diversas possibilidades.

Figura 19.42. Vários segmentos de reta.

Figura 19.43. *Jitter plot* com p-valores das comparações de médias.

Figura 19.44. Gráficos violino padrão, por grupos.

Figura 19.45. Gráficos violino com mudança de escala no eixo x.

Figura 19.46. Gráficos violino com mudança de escala no eixo x e y.

Figura 19.47. Gráficos violino com mudança de escala no eixo x e y e de preenchimento.

Figura 19.48. Gráficos violino com mudança de títulos, subtítulo, *caption*, título dos eixos e da legenda.

Figura 19.49. Gráficos violino com fontes modificadas.

Figura 19.50. Gráficos de pontos com escala de cores modificadas manualmente com nomes.

Figura 19.51. Gráficos de pontos com escala de cores modificadas manualmente em hexadecimal.

Figura 19.52. Gráficos de barras com paleta de cores do tipo Viridis.

Figura 19.53. Construindo paleta de cores com hcl.colors().

Figura 19.54. Temas nativos do ggplot.

Figura 19.55. gráfico de pontos com tema clássico.

Figura 19.56. Tema STATA do pacote ggthemes.

Figura 19.57. Tema "par", que se assemelha ao tema padrão do pacote graphics, nativo do R.

Figura 19.58. Gráfico de pontos com todos os dados.

Figura 19.59. Gráficos de pontos separados por grupos, horizontalmente.

Figura 19.60. Gráficos de pontos separados por grupos, verticalmente.

Figura 19.61. Gráficos de pontos separados por grupos entre linhas e colunas.

Figura 19.62. Gráficos de pontos separados por dois grupos subdivididos somente em colunas.

Figura 19.63. Definindo gráfico subdivididos em duas colunas.

Figura 19.64. Gráficos distintos em uma linha, formando uma só figura.

Figura 19.65. Gráficos distintos em duas linhas, formando uma só figura.

Figura 19.66. Gráficos distintos entre linhas e colunas, formando uma só figura.

Figura 19.67. Quatro gráficos distintos em uma só figura.

Figura 19.68. Quatro gráficos distintos dispostos em uma só coluna.

Figura 19.69. Quatro gráficos distintos com *layout* modificado.

Figura 19.70. Gráficos distintos com *layout* desenhado previamente.

Figura 19.71. Acrescentando espaços vazios na figura final.

Figura 19.72. Gráfico de barras simples criado pelo pacote graphics.

Figura 19.73. Gráfico de barras do graphics transformado em objeto ggplot.

Figura 19.74. Gráficos de pacotes originalmente de pacotes diferentes postos em uma só figura.

Figura 19.75. Gráficos violino sobrepostos a *boxplots*, sobrepostos a gráficos *jitter*.

Figura 19.76. Sobrepondo *boxplot* aos gráficos de pontos e avaliando visualmente distribuições de dados.

Figura 19.77. Mapa-múndi com base no ggplot.

Figura 19.78. Modificação das escalas e tema do mapa.

Figura 19.79. Modificação das cores de cada país no mapa.

Figura 19.80. Destaque de países específicos no mapa.

Figura 19.81. Países com valores de uma variável contínua diferentes.

Figura 19.82. Atribuindo paleta de cores de preenchimento ao mapa.

Figura 19.83. Paleta de cores em tons de cinza no mapa.

Figura 19.84. Mapa dos estados brasileiros em paleta vermelha de gradação.

Figura 19.85. Mapa de um estado brasileiro (Piauí) em paleta vermelha de gradação por município.

Figura 19.86. Gráfico de pontos gerado pelo graphics do R.

LISTA DE SIGLAS

AIC - Critério de informação de Akaike

AIQ - Amplitude interquartil

ANOVA - *ANalysis Of Variance*

AUC - *Area Under The Curve*

BIC - Critério de informação bayesiano.

COVID-19 - *Coronavirus Disease* 2019

CPF - Cadastro de Pessoa Física.

CRAN - *Comprehensive R Archive Network*

CSV - *Comma-Separated Values*.

CV - Coeficiente de variação.

DM – Diabetes melitos.

ELISA - *Enzyme-Linked Immunosorbent Assay*.

FN – Falso negative.

FP – Falso positive.

GG - Greenhouse-Geisser.

HDL - *High-density Lipoprotein*.

HF - Huynh-Feldt.

HIV - *Human Immunodeficiency Virus*.

HPV - *Human Papillomavirus* (Papilomavírus Humano)

IBGE - Instituto Brasileiro de Geografia e Estatística

IC - Intervalo de Confiança.

IDE - *Integrated Development Environment*.

IgG - *Immunoglobulin G*.

IMC – Índice de Massa Corpórea

IPEA - Instituto de Pesquisa Econômica Aplicada.

IQR - Interquartile Range (Intervalo Interquartil)

K-S - Kolmogorov-Smirnov

MAE - Mean Absolut Error (erro absoluto médio)

NA - Not Available.

OR – Odds Ratio.

PAS - Pressão Arterial Sistólica.

PCR – Proteína C Reativa

Q-Q – Quantile-Quantile

RDW - Red cell Distribution Width

RG - registro geral.

RMSE – Root Mean Squared Error (raiz quadrática média)

ROC - Receiver Operating Characteristic

RP – Razão de Prevalência

RR – Risco Relativo

RV- - Razão de verossimilhança negativa

RV+ - razão de verossimilhança positiva

SD – Standart desviation (desvio padrão)

TCC - Trabalho de Conclusão de Curso

TGP – Transaminase Pirúvica; mesmo que: ALT = Alanina Aminotransferase.

TSV - tab-separated values.

UPA - Unidades de Pronto Atendimento

VCM (Volume Corpuscular Médio

VIF - Variância da Inflação do Fator.

VN – Verdadeiro negativo.

VP – Verdadeiro positive.

VPN - Valor preditivo negativo.

VPP - Valor preditivo positivo.

ÍNDICE REMISSIVO

A

acaso · 293
aceitar H_0 · 119
acréscimo · 106
acurácia · 262
ajuda · 23
amodal · 77
amostra · 63
amostra aleatória estratificada · 65
amostra aleatória simples · 65
amostragem · 64
amplitude · 93
amplitude interquartil · 88, 95
análise de Variância · 199
ANOVA · 199, 255
ANOVA de medidas repetidas · 210
área sob a curva · 276
arrays · 18
AUC · 276

B

bancos de dados diferentes · 57
barras de erro · 373
betas · 145
bimodal · 77
binomial · 167
Bland-Altman · 226
Bonferroni · 204
boxplot · 87, 365

Ch

character · 33
cheatsheets · 24
Chi-Square Goodness of Fit Test · 172

C

classe · 14
classes · 13
coeficiente de correlação · 137
coeficiente de determinação · 149
coeficiente de variação · 101
Cohen · 223
combinar · 59
comparar modelos de regressão · 156
Comprehensive R Archive Network · 2
concordância · 223
console · 5
cores · 393
correção de Yates · 176
correlação · 133, 242
correlação negativa · 135
correlação positiva · 134
covariáveis · 161
CRAN · 22
critério de informação bayesiano · 157
critério de informação de Akaike · 157
curva gaussiana · 122
curva normal · 122
curva ROC · 273
cutoff · 273

D

D'Agostino · 126
dados estruturados · 29
dados faltantes · 46
dados ordenados · 75
data frame · 19
decréscimo · 106
desvio padrão · 99, 372
desvio padrão dos resíduos · 149
diagramas de caixa · 365
diagramas de dispersão · 134
dicotomização · 174
distribuição de dados · 121
distribuição normal · 121
double · 15
dplyr · 42
dummy · 57, 166
Duncan · 204
Dunnett · 204

E

editor · 4
endemia · 301
ensaio clínico randomizado · 309
ensaio comunitário · 310
ensaio de campo · 309
Environment · 6, 42, 61
epidemia · 300
epidemiologia · 293
erro padrão da média · 372
erro tipo I · 237
erro tipo II · 237
erros de amostragem · 64
escalas · 385
escore Z · 151
esfericidade · 211
especificidade · 263
estratos · 65
estudo caso-controle · 306
estudo de coorte · 307

estudo ecológico · 304
estudo transversal · 305
estudos experimentais · 308
estudos observacionais descritivos · 302
evento · 105
exportar · 39

F

facetar · 402
FALSE · 11
fator multiplicativo · 108
fatores · 14
filtrar dados · 53
funções · 20

G

ggplot2 · 415
gráfico de área · 360
gráfico de densidade · 347
gráfico de linha · 358
gráfico de pontos · 353
gráfico de pontos sobrepostos · 368
gráficos · 340
gráficos de barras · 362
gráficos de tendência · 356
gráficos violinos · 369
graus de liberdade · 115

H

H_0 · 111
H_1 · 111
H_a · 111
head · 44
heterocedasticidade · 202
histograma · 343
homocedasticidade · 202

I

IDE · 3
IMC · 55
import · 37
importar · 39
imputação de dados · 51
incidência · 297
inclinação · 144
input · 39
instale o RStudio · 3
inteligência artificia · 25
intercepto · 144
intervalo de confiança da média · 372
intervalo interquartil · 95

J

juntar · 59

K

kappa · 223
kappa de Cohen · 224
Kolmogorov-Smirnov · 126

L

linguagem R · 1
lista · 17

M

mapas · 420
matriz · 18
máximo · 93
média · 73
média geométrica · 73
média harmônica · 73
mediana · 75
médias independentes · 252
médias pareadas · 254
mínimo · 93
moda · 76
modelo · 148
multicolinearidade · 162
multimodal · 77

N

n amostral · 235
nível de significância · 116
nomear variáveis · 34
normalização · 152
numeric · 15, 33
número · 27
números inteiros · 15

O

odds ratio · 166, 312
operadores lógicos · 10
operadores matemáticos · 10
outliers · 87
output · 39

P

padrão-ouro · 259

paletas de cores · 395
pandemia · 300
Pearson · 137
percentis · 85
pergunta biológica · 204
pergunta científica · 63
perguntas biológicas · 235
pipe · 53
poder de teste · 251
ponto de corte · 285
população · 63
post hoc · 204
predizer · 148
prevalência · 267, 294
Prevalência de período · 296
Prevalência na vida · 296
Prevalência pontual · 296
probabilidade · 105
p-valor · 119
p-value · 118

Q

Q-Q plot · 130, 349
quadro · 327
qualidade · 27
quartis · 86
qui-quadrado · 171, 239
qui-quadrado de aderência · 173
qui-quadrado de independência · 173

R

R · 1
R^2 ajustado · 150
raiz quadrada · 20
razão de chance · 166, 168, 312
razão de prevalência · 312, 314
razão de verossimilhança negativa · 266
razão de verossimilhança positiva · 266
regressão · 144, 244
regressão linear simples · 144
regressão logística · 166, 248
regressão múltipla · 150
rejeitar H_0 · 112, 119
resíduos · 148
revisão sistemática · 310
RStudio · 3

S

salvar · 35, 432
Scheffe · 204
script · 8
seguimentos de reta · 382
seleção · 52
sensibilidade · 262
sequência · 21
Shapiro-Wilk · 126
Sidak · 204
SNK · 204
Spearman · 137
subamostragem · 69
surto · 299

T

tabela · 327
tabelas 2x2 · 259
tabulação · 32
tail · 44
tamanho da fonte · 390
taxa · 107
taxa de acréscimo · 108
taxa de decréscimo · 109
temas · 398
teste de Friedman · 216
teste de hipótese · 111
teste de Mauchly · 211
teste de normalidade · 127
teste de uma amostra · 184
teste de Wilcoxon para uma amostra · 187
teste de Wilcoxon pareado · 193
teste diagnóstico · 259
teste exato de Fisher · 176
teste Kruskal-Wallis · 202, 207
teste não paramétrico de Mann-Whitney · 189
teste não paramétrico de Wilcoxon para uma amostra · 187
teste t de *Studant* · 184, 190
teste t paramétrico para uma amostra · 185
teste t pareado · 193
testes não paramétricos · 131
testes paramétricos · 131
tibble · 19
tidyverse · 41
tipos de linhas · 357
títulos · 389
trimodal · 77
TRUE · 11
Tukey · 204
TXT · 35

U

unimodal · 77
universo amostral · 105
UTF-8 · 31

V

valor de kappa · 224
valor preditivo negativo · 264
valor preditivo positivo · 264
variância · 99
variância da inflação do fator · 162
variáveis categóricas · 29

variáveis qualitativas nominais · 28
variáveis quantitativas contínuas · 28
variável dependente · 145
variável independente · 145
variável quantitativa discreta · 28
vetores lógicos · 16
viés de tendência · 232
vírgula · 30
VPN · 264
VPP · 264

Y

Youden · 285

ANOTAÇÕES

Nesta obra, adentre o mundo da bioestatística e da epidemiologia de forma leve e desvende os segredos por trás dos números das ciências biológicas e da saúde. Este livro irá quebrar seus paradigmas e lhe mostrar que linguagem de programação não é um bicho de sete cabeças. Você aprenderá, passo a passo, como utilizar essa poderosa ferramenta de maneira fácil e acessível, mesmo sem ter qualquer experiência anterior com programação.

O R é gratuito, então, liberte-se das amarras de softwares caros e de origem duvidosa para realizar suas análises. Com essa linguagem, você terá em suas mãos um universo de possibilidades para explorar e analisar dados de forma eficiente e confiável. Ao longo desta jornada, você aprenderá a manipular e visualizar dados, realizar vários testes estatísticos, construir modelos preditivos, descobrir a melhor forma de apresentar seus resultados, seja em tabelas ou em gráficos. Tudo de maneira descomplicada, prática e objetiva.

Não importa se você é estudante, pesquisador ou profissional, "Princípios de bioestatística e epidemiologia: fundamentos práticos em R" fornecerá todo o conhecimento necessário para impulsionar seus estudos e conquistar bons resultados.

9 786501 507170